UNIVERSIDADE FEDERAL DE MINAS GERAIS

ESCOLA DE ENGENHARIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ESTUDO, PROJETO E IMPLEMENTAÇÃO DE UM MODELADOR DE SÓLIDOS VOLTADO PARA APLICAÇÕES EM ELETROMAGNETISMO

ANA LIDDY CENNI DE CASTRO MAGALHÃES

ORIENTADOR: PROF. DR. RENATO CARDOSO MESQUITA

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção de título de Doutor em Engenharia Elétrica

Área de Concentração: Automática

Linha de Pesquisa: Otimização e Projeto Assistidos por Computador

BELO HORIZONTE - SETEMBRO DE 2000

"Há pessoas estrelas e há pessoas cometas. Os cometas passam, desaparecem. Apenas são lembrados pelas datas em que passam e retornam. As estrelas permanecem. O sol permanece. Passam-se os anos, milhões de anos, e as estrelas permanecem.

Há muita gente cometa. Passa pela nossa vida apenas por instantes. Gente que não prende ninguém e que a ninguém se prende. Gente sem amigos, gente que passa pela vida sem iluminar, sem aquecer, sem marcar presença. (...) Ser cometa é não ser amigo. É ser companheiro apenas por instantes. É explorar os sentimentos humanos. A solidão é resultado de uma vida de cometa. Todos passam. E a gente também passa pelos outros.

Importante é ser estrela. Permanecer. Estar presente. Marcar presença. Estar junto. Ser luz. Ser calor. Ser vida. Amigo é estrela. Podem passar os anos, podem surgir distâncias, mas a marca fica no coração. Coração que não quer enamorar-se de cometas, que apenas atraem olhares passageiros. (...) Há necessidade de criar-se um mundo de estrelas e todos os dias poder contar com elas. Todos os dias ver a luz e sentir seu calor. Assim são os amigos estrelas da vida da gente. Pode-se contar com eles. Eles são presença. São coragem nos momentos de tensão. São luz nos momentos de escuridão. São segurança nos momentos de desânimo.

Ser estrela neste mundo passageiro, neste mundo cheio de pessoas cometas, é um desafio, mas acima de tudo, uma recompensa. É nascer e ter vivido, e não apenas ter existido."

(autor desconhecido)

Aos meus pais, Mário e Anna Maria, de mente sempre aberta ao novo, que intensificam em todos os momentos o brilho de qualquer satélite que venha pairar a seu redor, ainda que à custa de sua própria energia. Vocês são os meus maiores mestres !

Ao Breno, minha metade, pelo apoio incondicional, carinho, compreensão e incentivo durante mais esta etapa da minha formação.

Aos meus filhos, Breno Jr., Pedro e Celina, com quem aprendo mais sobre mim mesma a cada dia.

"Os que tiverem ensinado a muitos os caminhos da justiça brilharão com as estrelas, por toda a eternidade"
Dn, 12,3.

AGRADECIMENTOS

Ao Professor Renato Cardoso Mesquita, pela orientação, ensinamentos, atenção, competência, estímulo, exigência e disponibilidade durante o desenvolvimento deste trabalho.

Aos amigos Cássia Regina Santos Nunes, Christiane de Lisieux Leal e Mol, Cleverson Albert Shimizu, Kristian Magnani dos Santos, Raoni Maíra Resende e Sílvio Antônio Nunes, alunos de iniciação científica, que tanto contribuíram para a execução deste trabalho, pela dedicação, interesse e paciência.

Aos Professores João Antônio de Vasconcelos e Rodney Rezende Saldanha, do GOPAC – Grupo de Otimização e Projeto Assistidos por Computador, pelo apoio, ensinamentos e incentivo.

Ao Professor Christiano Gonçalves Becker, do Departamento de Ciência da Computação / ICEX – UFMG, pelos primeiros ensinamentos em computação, decisivos para que eu me dedicasse a essa área do conhecimento.

Aos colegas do GOPAC – Simone, Manuel, Lomônaco, Gustavo, Márcio, Henrique, Rogério, Antônia, Ana Paula e Sérgio – pelas discussões e troca de experiências, sempre proveitosas, que pudemos ter.

Aos colegas Maria Luiza, Marisa, Dora, Amilton, Marco Aurélio, Joaquim, José Hissa, Régis, Ernane, Zélia, Parma, Tarcísio, entre tantos outros, pelo convívio enriquecedor, cooperação e companheirismo demonstrados durante todo esse período. Ao colega Gustavo Guimarães Parma, pelo apoio e dedicação constante em manter a nossa estrutura de equipamentos em boas condições de uso.

Aos demais colegas – mestrandos e doutorandos – , professores e funcionários do CPDEE / UFMG que, de uma maneira ou de outra , colaboraram para o desenvolvimento deste trabalho.

A D. Carmen Lydia Ferolla de Castro Magalhães, sempre amiga e disponível, pela preciosa revisão do texto.

A Nilzete Rodrigues de Souza, ue por inúmeras vezes assumiu meu papel de mãe e de dona-de-casa.

À CAPES – Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior, pela concessão de Bolsa de Estudos durante o período de realização deste trabalho.

Ao CNPq – Centro Nacional de Desenvolvimento Científico e Tecnológico, pelo suporte financeiro ao projeto integrado "Novas Técnicas em Modelamento de Sólidos para Eletromagnetismo" (processo 520889/96-7 NV), à FAPEMIG – Fundação de Amparo à Pesquisa de Minas Gerais, pelo suporte financeiro ao projeto "Modelador de Sólidos aplicado ao Cálculo de Campos em Três Dimensões" (processo TEC 1425/97) e à FINEP – Financiadora de Estudos e Projetos, pelo suporte financeiro ao projeto "Engenharia de Software" do Programa RECOPE (processo 2986).

RESUMO

Este trabalho compreende o desenvolvimento de um modelador de sólidos baseado em técnicas avançadas de modelamento geométrico tridimensional, a ser acoplado a um gerador de malhas adaptativas de um pré-processador voltado para a resolução de problemas eletromagnéticos utilizando o Método de Elementos Finitos. O sistema desenvolvido permite construir modelos manufaturáveis, combinando primitivas CSG (*Constructive Solid Geometry*) instanciadas ou definidas por varredura, derivando a representação por fronteira (B-rep – *Boundary Representation*) a partir da representação CSG e gerando a malha superficial de elementos finitos sobre essa fronteira. O modelador traz como principal inovação uma estrutura de dados B-rep que permite representar fronteiras internas do modelo, garantindo a interpretação de maneira única da interface entre componentes e a compatibilidade da malha superficial de elementos finitos gerada. Permite também gerar primitivas complexas a partir de perfis compostos por mais de um contorno, o que reduz a necessidade de aplicação de operações booleanas. Além de facilitar e agilizar o processo de criação, edição, visualização e acesso à representação computacional de sólidos manufaturáveis, este modelador destina-se a gerar uma malha superficial de elementos finitos de boa qualidade, capaz de fornecer as informações necessárias para a simulação eletromagnética do modelo. O sistema é composto por quatro subsistemas principais: a Interface, responsável pela criação, edição e visualização de sólidos; o subsistema de Modelagem, que engloba as operações de manipulação da forma; o subsistema de Representação, ou Núcleo, que gerencia o acesso às estruturas de dados e o subsistema que cuida da Geração da Malha de Elementos Finitos sobre o modelo. Ele está sendo desenvolvido pelo GOPAC (Grupo de Otimização e Projeto Assistidos por Computador), no CPDEE / UFMG (Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica da Universidade Federal de Minas Gerais), utilizando os conceitos de orientação para objetos, o ambiente de desenvolvimento *Borland C++ Builder*, a biblioteca de estruturas genéricas de dados STL (*Standard Template Library*) e a biblioteca gráfica *OpenGL*. Os resultados obtidos podem ser armazenados em uma base de dados de formato neutro, possibilitando o intercâmbio de informações com outros aplicativos e certificando a potencialidade do modelador para diversas aplicações em engenharia.

ABSTRACT

This work covers the development of a solid modeler based on advanced techniques of tridimensional geometric modeling. It will be coupled to an adaptive mesh generator of a pre-processor, aiming at the solution of electromagnetic problems by means of finite-element method. Manufacturable models can be obtained through the combination of CSG (Constructive Solid Geometry) primitives directly instantiated or defined by sweep operations. The boundary (B-rep) representation as well as the finite-element surface mesh are derived from the CSG model. A new B-rep data structure represents the modeler main innovation, since it allows the handling of inner boundaries between the components of manufacturable B-rep models, guarantying the uniqueness of the inner boundary representation and assuring the finite-element compatibility for the generated mesh. This modeler also generates complex primitives by sweeping or swinging multi-contour profiles, which reduces the necessity for boolean operations. Besides supporting and speeding up the processes of creation, manipulation and visualization of manufacturable solid representations, the main goal of this modeler is to generate a good surface finite-element mesh, capable of providing the essential information for the electromagnetic simulation of the model. The modeler is composed of four main subsystems: the Interface, responsible for the user access on model creation, edition and visualization; the Modeling subsystem, enclosing the operations for shape manipulation; the Kernel, which manages and controls data structures access; the Mesh Generation subsystem, responsible for the creation and improvements of surface finite-element meshes. The modeler is being developed by the GOPAC – Optimization and Computer-Aided Design Group – at CPDEE / UFMG – Center of Research and Development on Electrical Engineering / Federal University of Minas Gerais. It uses object-oriented concepts, the *Borland C++ Builder* environment coupled to the *OpenGL* graphics library and the STL – *Standard Template Library*. The results obtained can be stored in a neutral format database, which will allow information interchange with other applications and assure the modeler potential for engineering applications.

SUMÁRIO

I – INTRODUÇÃO 1
I.1 – Um Breve Histórico da Evolução da Modelagem.......... 1
I.2 – Conceitos Básicos Relacionados: Modelo, Modelagem, CAD.......... 4
I.3 – A Modelagem Computacional de Sólidos.......... 6
I.3.1 – Requisitos para a Representação Computacional de Sólidos.......... 6
I.3.2 – Principais Esquemas de Representação 8
I.4 – O Método de Elementos Finitos e a Modelagem de Sólidos.......... 11
I.4.1 – Arquitetura Geral de Sistemas Baseados no Método de Elementos Finitos 11
I.4.2 – A Descrição da Geometria por Meio de um Modelador de Sólidos 13
I.4.3 – Técnicas de Discretização do Domínio 13
I.4.4 – A Atribuição de Características Físicas 16
I.5 – Motivações e Objetivos 17
I.6 – Organização do Trabalho.......... 19

II - DESENVOLVIMENTO DE UM MODELADOR DE SÓLIDOS VOLTADO PARA APLICAÇÕES EM ELETROMAGNETISMO 21
II.1 – Um Breve Histórico.......... 21
II.2 – Escopo deste Trabalho.......... 22
II.3 – Características Gerais de Projeto 23
II.3.1 – Especificação Funcional.......... 23
II.3.2 – Especificação de Requisitos 26
II.3.2.1 – Utilização de Programação Orientada para Objetos 26
II.3.2.2 – Intercâmbio de Dados com Outros Programas.......... 27
II.3.2.3 – Parametrização do Sistema 27
II.3.3 – A Análise Orientada para Objetos.......... 28
II.3.4 – Soluções Orientadas para Objetos.......... 29
II.4 – Características Gerais de Implementação 31
II.4.1 – A Importância da Independência Funcional entre os Módulos 32
II.4.2 – Regras Adotadas para a Obtenção de Programas Modularizados 33
II.5 – Características Gerais da Documentação Produzida.......... 34

III – O SUBSISTEMA DE INTERFACE 35
III.1 – Principais Características de Projeto 35
III.1.1 – Requisitos Relacionados ao Projeto de Interfaces 36
III.1.2 – Requisitos Relacionados à Interface do Modelador.......... 38
III.1.3 – O Modelamento Obtido 40
III.1.4 – Soluções Orientadas para Objetos 43
III.2 – Principais Características de Implementação 44
III.2.1 – Integração entre o Builder e a OpenGL 44
III.2.2 – O Ambiente e o Controle Geral do Sistema.......... 44
III.2.3 – Tratamento de Comandos 47
III.2.3.1 – O Fornecimento de Parâmetros 48
III.2.4 – Manipulação da Área de Trabalho.......... 49
III.2.5 – Recursos para Visualização 50
III.2.6 – A Visualização Estéreo.......... 51
III.2.7 – Definição e Utilização de Cores 53
III.2.8 – A Definição de *Labels* 53
III.2.9 – Apresentação de Resultados 54
III.3 – Principais Componentes da Interface 55
III.3.1 – Principais Formulários e Caixas de Diálogo Disponíveis 55
III.3.2 – Principais Comandos Disponíveis 58

IV – O SUBSISTEMA DE MODELAGEM 59
IV.1 – O Modelamento CSG 59
IV.1.1 – Propriedades da Representação CSG 60
IV.1.2 – A Estrutura de Dados CSG Padrão 61
IV.1.3 – Variações da Estrutura CSG de Interesse para o Eletromagnetismo 63
IV.2 – Principais Características de Projeto 64
IV.2.1 – Requisitos Relacionados à Modelagem 64
IV.2.2 – O Modelamento Obtido 66
IV.2.3 – Soluções Orientadas para Objetos 70
IV.3 – Principais Características de Implementação 72
IV.3.1 – Os Elementos Geométricos Básicos 73
IV.3.2 – Primitivas Disponíveis 74
IV.3.2.1 – Primitivas Planares 74
IV.3.2.2 – Primitivas Sólidas 75
IV.3.3 – Operações de Varredura 78
IV.3.3.1 – A Varredura Translacional 79
IV.3.3.2 – A Varredura Rotacional 80
IV.3.3.3 – Considerações sobre Operações de Varredura 81
IV.3.4 – As Transformações Geométricas 82
IV.3.5 – Operações Booleanas 84
IV.3.5.1 – A Necessidade da Regularização 86
IV.3.5.2 – A Interseção Regularizada 88
IV.3.5.3 – A União Regularizada 90
IV.3.5.4 – A Diferença Regularizada 91
IV.3.6 – A Operação de Montagem 91
IV.3.6.1 – Tratamento de Fronteiras Internas 92
IV.3.7 – A Definição de Perfis Compostos 97
IV.3.8 – A Estrutura CSG Utilizada 98

V – O SUBSISTEMA DE REPRESENTAÇÃO 99
V.1 – O Modelamento B-rep 99
V.1.1 – Propriedades da Representação B-rep 100
V.1.2 – Estruturas de Dados B-rep 101
V.1.2.1 – Estruturas de Dados para Sólidos de Variedade Bidimensional 102
V.1.2.2 – Estruturas de Dados para Sólidos de Variedade Múltipla 103
V.1.3 – Construção de Sólidos B-rep pelos Operadores de Euler 106
V.1.3.1 – Operadores de Euler Aplicados a Sólidos de Variedade Bidimensional 106
V.1.3.2 – Operadores de Euler Aplicados a Sólidos de Variedade Múltipla 107
V.1.4 – A Validação de Modelos B-rep 109
V.1.5 – Inclusão de Superfícies Curvas 109
V.1.6 – O Processo de Avaliação da Fronteira 110
V.2 – Principais Características de Projeto 111
V.2.1 – Requisitos Relacionados à Representação Interna ou Núcleo 112
V.2.2 – A Estrutura B-rep Definida para este Modelador 113
V.2.3 – Operadores de Euler a serem Utilizados pela Estrutura B-rep Definida 114
V.2.4 – O Modelamento Obtido 118
V.2.5 – Soluções Orientadas para Objetos 122
V.3 – Principais Características de Implementação 124
V.3.1 – Estruturas de Dados Auxiliares 124
V.3.2 – A Estrutura B-rep 125
V.3.3 – Operadores de Euler 127
V.3.3.1 – MVFR e KVFR 127
V.3.3.2 – MEV e KEV 128
V.3.3.3 – MEF e KEF 129
V.3.3.4 – KEML e MEKL 130

V.3.3.5 – MFKLH e KFMLH 131
V.3.3.6 – MSKR e KSMR 132
V.3.3.7 – MFFR e KFFR 133
V.3.3.8 – Operadores Adicionais GR, UR, GF, e UF 134
V.3.3.9 – Operadores Adicionais AF, DF, AR, e DR 135
V.3.4 – A Avaliação da Fronteira 136
V.3.4.1 – Geração da Malha sobre as Primitivas 137
V.3.4.2 – Interseção das Cascas dos Operandos Sólidos 138
V.3.4.3 – Classificação dos Componentes 140
V.3.4.4 – Avaliação Booleana e Eliminação dos Componentes Indesejados 143

VI – O SUBSISTEMA DE GERAÇÃO DE MALHA 145
VI.1 – Técnicas de Geração de Malha Utilizadas como Referência 145
VI.2 – Principais Características de Projeto 147
VI.2.1 – Requisitos Relacionados à Geração de Malha 147
VI.2.2 – O Modelamento Obtido 148
VI.2.3 – Soluções Orientadas para Objetos 151
VI.3 – Principais Características de Implementação 152
VI.3.1 – A Geração de Malha para a Varredura Rotacional 152
VI.3.1.1 – A Geração de Malha para a Varredura Rotacional de um Perfil Aberto 152
VI.3.1.2 – A Geração de Malha para a Varredura Rotacional de um Perfil Fechado 154
VI.3.1.3 – A Geração dos Operadores de Euler a partir da Varredura Rotacional 157
VI.3.2 – A Geração de Malha para a Varredura Translacional 158
VI.3.2.1 – A Geração de Malha para a Varredura Translacional Simples 158
VI.3.2.2 – A Geração de Malha para a Varredura Translacional Cônica 159
VI.3.2.3 – A Geração dos Operadores de Euler a partir da Varredura Translacional 161
VI.3.3 – A Geração de Malha sobre Faces Planares 161
VI.3.3.1 – O Programa *Triangle* 162
VI.3.3.2 – A Geração dos Operadores de Euler a partir dos Resultados obtidos do *Triangle* 164
VI.3.4 – Atribuição de *Label* e Especificação de Materiais 166
VI.3.5 – A Geração da Malha Volumétrica 168

VII – A GERAÇÃO DE MODELOS E DE BASE DE DADOS NEUTRA 169
VII.1 – O Intercâmbio de Dados e a Geração da Base de Dados Neutra 169
VII.1.1 – Vantagens e Desvantagens da Utilização de uma Base de Dados Neutra 169
VII.1.2 – Esforços Internacionais em Busca de uma Padronização 170
VII.1.3 – A Base de Dados Desenvolvida pelo GOPAC 172
VII.2 – Alguns Exemplos Básicos 172
VII.2.1 – A Geração de um Bloco 173
VII.2.2 – A Geração de Sólidos Ocos 174
VII.2.3 – A Geração de Sólidos Vazados 175
VII.2.4 – A Colagem e a Montagem de Sólidos 176
VII.3 - A Geração de Modelos B-rep Simples 178
VII.3.1 – Um Modelo em Forma de L 179
VII.3.2 – Variações na Densidade da Malha 184
VII.3.3 – A Inclusão de Buracos 185
VII.4 –Exemplos aplicados ao Eletromagnetismo 186
VII.4.1 – O Modelamento de Cabos 186
VII.4.2 – O Modelamento de Máquinas Elétricas 188
VII.4.3 – O Modelamento de Isoladores 192
VII.4.4 – O Modelamento de Bobinas 194
VII.4.5 – O Modelamento de Transformadores 195
VII.4.6 – Outros Modelos de Interesse 198

VIII – CONCLUSÃO 199
VIII.1 – Reflexões Acerca deste Trabalho, seu Contexto e Resultados 199
VIII.2 – Situação Atual do Modelador 202
VIII.3 – Contribuições deste Trabalho 205
VIII.4 – Sugestões para Futuros Trabalhos 208

APÊNDICE I – PADRONIZAÇÃO ADOTADA DURANTE O DESENVOLVIMENTO DO MODELADOR........ 211
AI.1 – A Documentação de Módulos 211
AI.2 – A Documentação de Programas 214
AI.3 – A Documentação Global do Modelador........ 216

APÊNDICE II – COMANDOS DISPONÍVEIS........ 221

REFERÊNCIAS BIBLIOGRÁFICAS........ 227

LISTA DE FIGURAS

I.1 – Integração CAD/CAM: da definição gráfica à produção 5
I.2 – Elucidação do conceito de sólido e de variedades simples e múltipla 7
I.3 – Arquitetura híbrida ideal 10
I.4 – Tarefas executadas pelo pré-processador, processador e pós-processador 12
I.5 – Elucidação do conceito de compatibilidade na discretização 14
I.6 – Regularização de uma triangulação por meio do deslocamento do nó central para o baricentro do polígono 16

II.1 – Estrutura funcional do modelador 25
II.2 – A participação do arquivo neutro no intercâmbio de dados entre as etapas da análise de um problema 27
II.3 – Estruturação de assuntos do modelador em desenvolvimento 29
II.4 – O padrão *Iterator* fornecendo uma interface uniforme para manipular diferentes estruturas de dados 30
II.5 – Redefinição de operações no padrão de projeto para *Template* 31

III.1 – Diagrama das principais classes e relações de dependência entre elas 40
III.2 – Identificação de assuntos relacionados ao Subsistema de Interface 42
III.3 – Visão geral da estrutura de dados *Command* projetada para encapsular o tratamento de Comandos 43
III.4 – Tela principal do modelador 45
III.5 – Representação gráfica da hierarquia de menus do modelador 46
III.6 – Formas alternativas de apresentação de um modelo 51
III.7 – Interface do programa Vimesh3D 52
III.8 – A edição de tabelas acoplada à criação e edição de *labels* 54
III.9 – Caixas de diálogo para abertura e gravação de arquivos 55
III.10 – Caixas de diálogo para definição e movimentação de um plano de trabalho 55
III.11 – Caixas de diálogo para especificação de preferências do usuário e configuração de impressora 56
III.12 – Caixas de diálogo para a definição de cores 56
III.13 – Formulário para edição de tabelas contendo características físicas 56
III.14 – Caixas de diálogo para a definição de cor e espessura de linha 56
III.15 – Caixas de diálogo para a definição dos atributos corrente e tensão 57
III.16 – Caixas de diálogo para a definição de condições de contorno e materiais 57
III.17 – Caixas de diálogo para fornecer ordem de integração, enrolamento e tipo de *label* a ser editado 57
III.18 – Caixas de diálogo para inclusão e edição de labels 57

IV.1 – Exemplos de árvores CSG e a melhoria obtida com a inclusão da transformação geométrica 60
IV.2 – Diagrama das principais classes e relações de dependência entre elas 67
IV.3 – Identificação de assuntos relacionados ao Subsistema de Modelagem 70
IV.4 – Relações de dependência na estrutura *Primitive3D* 71
IV.5 – Relações de dependência na estrutura CSG-Model 71
IV.6 – O modelo de projeto para composição recursiva representando árvores CSG 72
IV.7 – Exemplo de degeneração em uma varredura genérica 82
IV.8 – Passos para a realização da interseção regularizada 87
IV.9 – Operações booleanas em sólidos tridimensionais 88
IV.10 – Candidatos de uma interseção booleana regularizada e o teste da fronteira 89
IV.11 – Componentes candidatos de uma união booleana regularizada 90
IV.12 – Componentes candidatos de uma diferença booleana regularizada 91
IV.13 – Elucidação do conceito de *s-set* 93
IV.14 – A união e a montagem de dois *s-sets* 94
IV.15 – Exemplos de operações booleanas e de montagem sobre *s-sets* 97
IV.16 – Elucidação do conceito de perfil composto 97

V.1 – Características gerais da representação B-rep 100
V.2 – A topologia do cubo representada por modelo planar: cubo maciço e cubo vazado 102
V.3 – A estrutura de dados Semi-aresta (*Half-Edge*) 103
V.4 – Hierarquia da estrutura de dados para sólidos de variedade múltipla 104
V.5 – Estrutura de dados para sólidos B-rep de variedade múltipla proposta por Lee 105
V.6 – Perda de informação devido a aproximação 110
V.7 – Características da estrutura B-rep proposta: uso-face, uso-ciclo, uso-aresta, semi-aresta e uso-vértice 113
V.8 – Elucidação do conceito de uso-face e seus derivados 114
V.9 – Alguns exemplos de contagem de elementos 115
V.10 – Exemplos de divisão de cascas utilizando o compartilhamento de fronteiras 116
V.11 – Diagrama das principais classes e relações de dependência entre elas 119
V.12 – Identificação de assuntos relacionados ao Subsistema de Representação 122
V.13 – Relações de dependência da estrutura *BREP-Model* 123
V.14 – A estrutura de lista utilizada pela representação B-rep 123
V.15 – A estrutura de dados B-rep implementada nesse trabalho 125
V.16 – Efeito obtido ao serem aplicados os operadores MVFR e KVFR 127
V.17 – Formas de utilização das operações MEV e KEV 128
V.18 – Formas de utilização das operações MEF e KEF 130
V.19 – Formas de utilização das operações KEML e MEKL 131
V.20 – Efeito obtido ao serem aplicados os operadores KFMLH e MFKLH 132
V.21 – Representação gráfica das operações MSKR e KSMR 133
V.22 – Representação gráfica das operações MFFR e KFFR 133
V.23 – Efeito obtido ao serem aplicados os operadores *Glue Regions* e seu inverso, *Unglue Region* 134
V.24 – Efeito obtido ao serem aplicados os operadores *Glue Faces* e seu inverso, *Unglue Face* 134
V.25 – Efeito obtido ao serem aplicados os operadores *Assemble Faces*, *Assemble Regions* e seus inversos 136
V.26 – Fluxograma para as etapas da avaliação incremental da fronteira 137
V.27 – Interseção de faces não coplanares: inserção de segmentos em modelos poligonais e triangularizados 140
V.28 – Casos possíveis na interseção de dois triângulos e inserção de segmentos de forma a manter a triangulação 141
V.29 – Utilização do *raytrace* para a classificação de vértices 141
V.30 – Classificação e avaliação das operações booleanas 143

VI.1 – Exemplo da técnica CSG com geração de malha sobre primitivas 145
VI.2 – Recursividade na discretização da esfera e problemas com a subdivisão adaptativa 146
VI.3 – Identificação de assuntos relacionados ao Subsistema de Geração de Malha 149
VI.4 – Diagrama das principais classes e relações de dependência entre elas 151
VI.5 – Exemplo de varredura rotacional de um perfil aberto 153
VI.6 – Exemplo de traçado por nível da varredura rotacional de um perfil aberto 154
VI.7 – Exemplos de malhas obtidas pela varredura rotacional de um perfil aberto 154
VI.8 – Decomposição de um toro e possíveis discretizações geradas sobre a decomposição 155
VI.9 – Formas de geração de malha utilizando a varredura rotacional de um perfil fechado 156
VI.10 – Exemplo de traçado por nível da varredura rotacional de um perfil fechado 156
VI.11 – Exemplos de malhas obtidas pela varredura rotacional de um perfil fechado 157
VI.12 – Traçado de malha utilizado na varredura translacional simples 159
VI.13 – Exemplos de malhas obtidas pela varredura translacional simples 159
VI.14 – Traçado utilizado na varredura translacional cônica 160
VI.15 – Malha resultante da varredura translacional cônica de um perfil retangular 160
VI.16 – Exemplos de malhas obtidas pela varredura translacional 160
VI.17 – Variações de malha que podem ser obtidas pelo programa *Triangle* 162
VI.18 – Estratégia inicial para definir operadores de Euler a partir da malha gerada pelo *Triangle* 165
VI.19 – Exemplos da estratégia para definir os operadores de Euler que incluem no B-rep a malha gerada pelo *Triangle* 165
VI.20 – Explicação do funcionamento e implementação do mecanismo de *label* 167

VII.1 – Seqüência de operações aplicadas na construção de um bloco ... 173
VII.2 – Seqüência de operações aplicadas na geração de sólidos ocos ... 174
VII.3 – Seqüência de operações aplicadas na geração de sólidos vazados ... 175
VII.4 – Seqüência de operações aplicadas na união de regiões e geração de montagens ... 177
VII.5 – Etapas na geração do modelo em forma de L ... 179
VII.6 – Variações de modelos gerados a partir do perfil em forma de L ... 180
VII.7 – Variações na densidade de malha devido à divisão do segmento padrão em mais elementos ... 184
VII.8 – Mais exemplos de variações na densidade de malha e na forma da varredura ... 184
VII.9 – Variações na densidade de malha devido à manipulação do número de pontos sobre o perfil gerador. 184
VII.10 – Modelos obtidos com a varredura translacional de perfis compostos ... 185
VII.11 – Modelos obtidos com a varredura rotacional de perfis compostos ... 185
VII.12 – O modelamento de um cabo simples ... 186
VII.13 – O modelamento de um cabo composto por 3 fios ... 187
VII.14 – Árvore CSG para a construção de um motor de corrente contínua simplificado ... 188
VII.15 – Construção do motor anterior a partir de perfis compostos ... 188
VII.16 – Malha superficial resultante para um motor de corrente contínua simplificado ... 189
VII.17 – Compatibilidade existente entre as malhas obtidas sobre os perfis geradores ... 189
VII.18 – Malha superficial obtida para o primeiro exemplo de máquina de corrente contínua de dois pólos ... 190
VII.19 – Malha superficial obtida para o segundo exemplo de máquina de corrente contínua de dois pólos ... 191
VII.20 – Exemplo de malha superficial obtida para um isolador cerâmico de alta tensão ... 192
VII.21 – Exemplo de malha superficial obtida para um isolador em forma de pino ... 193
VII.22 – Exemplo de malha superficial obtida para um isolador específico para atmosfera contaminada ... 193
VII.23 – Exemplo de malha superficial obtida para um conjunto bobina-núcleo ... 194
VII.24 – Exemplo de malha superficial obtida para um conjunto bobina – núcleo em forma de cruz ... 194
VII.25 – Exemplo de árvore CSG para a construção de um transformador bifásico ... 195
VII.26 – Geração do núcleo por varredura translacional de um perfil composto ... 196
VII.27 – Processo de geração de malha empregado na construção do transformador ... 196
VII.28 – Detalhamento da colagem e montagem de faces ... 197
VII.29 – Visão geral do transformador ... 197
VII.30 – Malha superficial obtida para um divisor de tensão ... 198

LISTA DE QUADROS

I.1 – As quatro gerações de sistemas CAD 3
I.2 – Propriedades necessárias para classificar um objeto como sólido abstrato válido 6
I.3 – Propriedades desejáveis em um esquema de representação 8
I.4 – Principais esquemas de representação de sólidos 9
I.5 – Organizações encontradas em softwares de elementos finitos 12
I.6 – Principais métodos de geração automática de malha 15

III.1 – Principais características das classes relacionadas 41
III.2 – Máquina de estados finitos para o comando "*Sphere*" 47
III.3 – Arquivos disponibilizados pelo sistema 55
III.4 – Principais comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas 58

IV.1 – Parâmetros quantitativos associados a uma árvore 61
IV.2 – Regras gramaticais BNF para uma árvore CSG padrão 62
IV.3 – Estrutura dos nós em uma árvore CSG padrão 62
IV.4 – Formas de caminhamento em uma árvore binária 63
IV.5 – Principais características das classes relacionadas 68
IV.6 – Elementos geométricos básicos definidos 73
IV.7 – Primitivas planares disponíveis 74
IV.8 – Primitivas sólidas disponíveis 75
IV.9 – Parâmetros utilizados na definição de primitivas 77
IV.10 – Tipos de varredura translacional 79
IV.11 – Principais características da varredura translacional 79
IV.12 – Tipos de varredura rotacional 80
IV.13 – Principais características da varredura rotacional 80
IV.14 – Transformações geométricas de interesse 82
IV.15 – Matrizes de transformação no plano e no espaço 84
IV.16 – Propriedades da teoria dos conjuntos 85
IV.17 – Definição dos operadores a serem aplicados sobre *s-sets* 95

V.1 – Operadores de Euler propostos por Mäntylä e Sulomen 107
V.2 – Operadores de Euler para a construção de sólidos de variedade múltipla 108
V.3 – Operadores de Euler a serem utilizados nesse trabalho 117
V.4 – Principais características das classes relacionadas 120
V.5 – Descrição sucinta do conteúdo dos principais componentes 126
V.6 – Parâmetros utilizados nas operações MVFR e KVFR 128
V.7 – Parâmetros utilizados nas operações MEV e KEV 129
V.8 – Parâmetros utilizados nas operações MEF e KEF 130
V.9 – Parâmetros utilizados nas operações KEML e MEKL 131
V.10 – Parâmetros utilizados nas operações MFKLH e KFMLH 132
V.11 – Parâmetros utilizados nas operações MSKR e KSMR 133
V.12 – Parâmetros utilizados nas operações MFFR e KFFR 133
V.13 – Parâmetros utilizados nas operações GR, UR, GF e UF 135
V.14 – Parâmetros utilizados nas operações de montagem AF e DF 136
V.15 – Processo para cálculo da interseção entre as cascas 138
V.16 – Classificação das arestas por seus pontos extremos 142
V.17 – Classificação das faces em função de suas arestas 142
V.18 – Tabela de decisão das operações booleanas 144

VI.1 – Principais características das classes relacionadas 150
VI.2 – Seqüência de operadores de Euler gerada em cada etapa da varredura rotacional 158
VI.3 – Seqüência de operadores de Euler gerada em cada caso da varredura translacional 161
VI.4 – Formato dos arquivos com extensão ".POL" 163
VI.5 – Formato dos arquivos ".ELE" e ".NOD" 164
VI.6 – Opções de execução do *Triangle* controladas pelo modelador 164
VI.7 – Composição de dados de cada atributo utilizado para *labels* 167

VII.1 – Operações de Euler aplicadas na construção de um bloco 174
VII.2 – Operações de Euler aplicadas na geração de um sólido oco 175
VII.3 – Operações de Euler aplicadas na geração de um sólido com vazamento 176
VII.4 – Operações de Euler aplicadas na geração de uma montagem 178
VII.5 – Operações de Euler aplicadas na geração das faces laterais do modelo em forma de L – arquivo ".brp" 179
VII.6 – Operações de Euler aplicadas na geração das faces inferior e superior do modelo em forma de L – arquivo ".brp" 180
VII.7 – Arquivo neutro de geometria gerado para o modelo em forma de L 181
VII.8 – Arquivo neutro de malha gerado para o modelo em forma de L 182
VII.9 – Arquivo ".3DF" para visualização estéreo e/ou das faces do modelo em L 183

AI.1 – Esqueleto do módulo de definição 211
AI.2 – Esqueleto do módulo de implementação 212
AI.3 – Exemplos de uso de estruturas *ifdef's* e *ifndefs* para flexibilizar o programa 213
AI.4 – Cabeçalho para início de um arquivo 214
AI.5 – Rodapé para finalização de um arquivo 214
AI.6 – Formato básico para o cabeçalho de uma classe 214
AI.7 – Formato básico para o cabeçalho de um método ou função no arquivo ".cpp" 215
AI.8 – Fomato básico para documentar um método ou função no arquivo-cabeçalho 215
AI.9 – Formato básico para atributos de classe e variáveis diversas 215
AI.10 – Formato básico para tipos de dados e estruturas 215
AI.11 – Gabarito para a documentação do sistema como um todo 216
AI.12 – Gabarito para a documentação de sub-sistemas 216
AI.13 – Gabarito para a documentação de assuntos dentro de cada sub-sistema 216
AI.14 – Gabarito para a documentação de módulos 216
AI.15 – Gabarito para a documentação de arquivos de dados a serem manipulados pelo sistema 217
AI.16 – Gabarito para a documentação de estruturas de dados 218
AI.17 – Gabarito para a documentação de classes 218
AI.18 – Gabarito para a documentação de atributos de classe 218
AI.19 – Gabarito para a documentação de métodos de classes e funções 219
AI.20 – Gabarito para a documentação de tipos de dados definidos pelo usuário 219
AI.21 – Gabarito para a documentação de informações gerenciais sobre o arquivo 220
AI.22 – Gabarito para a documentação de menu e sub-menus 220
AI.23 – Gabarito para a documentação de comandos implementados no modelador 220

AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas 221

"A ciência sem a fé conduz à dúvida; a fé sem a ciência conduz à superstição.
As duas reunidas dão a certeza, e para uni-las não se deve jamais confundi-las"
Eliphas Lévi

I – INTRODUÇÃO

No contexto de Engenharia Assistida por Computador, um modelo é uma representação computacional de um objeto, que pode ser uma réplica de um produto real existente ou o projeto de um produto a ser construído. O nível de detalhe e informações do modelo deve ser definido levando-se em consideração as necessidades impostas pela aplicação, procurando-se abstrair apenas as informações relevantes do objeto visando à aplicação.

A opção pela construção de um modelo decorre de uma questão de conveniência: é freqüentemente mais fácil, mais prático e mais econômico analisar um modelo do que testar, medir ou experimentar utilizando objetos reais. Além das vantagens de análise, o modelo é também útil, se não necessário, como forma de transportar informação entre as etapas de concepção e obtenção do produto. Seja para o desenvolvimento de novos produtos ou modificação dos existentes, a presença do modelo é fundamental na resolução de questões referentes a projeto, análise e manufatura.

A construção, manipulação e acesso à representação de objetos em um computador é geralmente realizada por meio de uma ferramenta gráfica específica, tecnicamente denominada **sistema de modelagem** ou **sistema CAD** e mais popularmente conhecida por **modelador**. Se o objetivo é modelar objetos sólidos, torna-se necessário utilizar **modeladores de sólidos**. Este trabalho apresenta os resultados já obtidos, bem como os estudos realizados visando ao desenvolvimento de um modelador de sólidos a ser utilizado como pré-processador na resolução de problemas eletromagnéticos empregando o Método de Elementos Finitos. Um resumo da evolução das técnicas de modelagem, bem como os principais conceitos relacionados à representação computacional de sólidos e ao Método de Elementos Finitos introduzem o assunto. Dando continuidade, são apresentadas as bases do desenvolvimento deste trabalho, seus objetivos e sua estrutura geral.

I.1 – UM BREVE HISTÓRICO DA EVOLUÇÃO DA MODELAGEM

Pode-se considerar como datado de 1963 o início do uso de computadores no projeto e manufatura de produtos industriais. Desenvolvidos para as indústrias automobilística e aeroespacial, os primeiros sistemas CAD simplesmente modelavam curvas e superfícies isoladas. Os métodos utilizados então, provenientes de estudos matemáticos desenvolvidos em fins dos anos 50 e início dos anos 60, demandavam recursos computacionais consideráveis, escassos para a época, inviabilizando a construção de sistemas de projeto interativos.

No fim dos anos 60, sistemas computadorizados de projeto estavam disponíveis para produzir desenhos semelhantes aos desenhos manuais, compostos principalmente por linhas retas e arcos de circunferências e carregados de anotações como dimensões, notas e outros tipos de informação. Utilizavam vídeo gráfico em substituição às folhas de papel, evitavam muito do trabalho repetitivo, com mais eficiência.

No início dos anos 70, a associação de informação sobre profundidade com as linhas de um desenho 2D permitiu a definição de uma classe de objetos conhecida como objetos semi-tridimensionais (2½D) [Pra90]. O computador pôde, então, gerar uma representação unificada do objeto, ao invés das visões ortogonais tradicionalmente desenhadas, originando um modelo computacional do produto e não desenhos isolados. Generalizando essa idéia, surgiram os modelos realmente 3D – conhecidos como modelos fio-de-arame (*wireframe*) [Fol90] – , que possibilitaram gerar automaticamente desenhos do objeto de qualquer ponto de vista e em qualquer projeção escolhida pelo usuário, inclusive as projeções ortogonais. A falta de informação sobre superfícies do modelo impossibilitava a geração automática de visões com linhas e superfícies ocultas, exigindo o traçado de todas as arestas, o que congestionava o desenho, dificultando sua compreensão e conduzindo a interpretações ambíguas.

Progredindo rapidamente na década de 70 [Req82, Bad78], a modelagem de sólidos reuniu as vantagens dos modeladores fio-de-arame e dos de superfície, permitindo gerar modelos com informações referentes a arestas, faces e superfícies limitantes, bem como detalhes sobre a conectividade entre estes elementos. Além de gerar automaticamente imagens sob qualquer ponto de vista com linhas ou superfícies não visíveis, tornou possível computar volume, massa, momento de inércia e interferência entre componentes, o que é de grande valia para aplicações em engenharia.

Baseado em seus componentes modeláveis, os sistemas CAD podem ser divididos em quatro gerações [LaC95], como descrito no quadro I.1. Estas gerações de CAD formam uma hierarquia que permite relacionar os níveis de um projeto: de um modelo podem ser extraídas suas superfícies, e destas, as arestas, que podem ser projetadas em um plano, no qual se formam desenhos contendo vistas do modelo.

O aspecto mais significativo desta progressão é a ampliação da possibilidade de interpretação do modelo pelo computador. O desenho produzido manualmente era de interpretação exclusiva do projetista, engenheiro ou técnico. Mesmo quando um desenho similar era produzido usando um sistema de desenho 2D, a informação contida no computador não pretendia ter interpretação automática, sendo meramente uma codificação para o desenho a ser interpretado pelo homem. Com o modelo fio-de-arame aumentou-se a possibilidade de uso automático dos dados,

4a Geração – Sistemas CAD para Modelos Sólidos
– Objetos são representados pelo espaço 3D que eles ocupam.
– Características avançadas como cálculos de propriedade de massa e detecção de interferência podem ser realizadas automaticamente.

3a Geração – Sistemas CAD para Modelos de Superfície
– Objetos são representados pelo modelamento de superfícies selecionadas, não conectadas entre si e sem definir necessariamente volumes.
– Imagens realísticas iluminadas podem ser produzidas.

2a Geração – Sistemas CAD para Modelos Fio-de-Arame
– Arestas selecionadas são modeladas em 3D.
– Suporta a apresentação de linhas em 3D (fio-de-arame) e a geração automática de vistas 2D em relação a qualquer ponto de vista.

1a Geração – Sistemas CAD para Desenhos
– Objetos são representados pela projeção de arestas selecionadas em um plano 2D.
– Linhas e textos podem ser fornecidos e atualizados interativamente.
– Impressões podem ser geradas automaticamente.

Quadro I.1 – As quatro gerações de sistemas CAD [LaC95]

mas muitas aplicações eram ainda impossíveis devido à ausência de informações sobre as superfícies e da noção de interior e exterior. Modeladores de superfície permitiram a automatização de certas aplicações, mas representavam somente superfícies desconexas, não definindo, necessariamente, volumes. Isto pode ser observado na figura do quadro I.1: conversões *top-down* podem ser usadas para extrair progressivamente dos modelos sólidos informações geométricas sobre suas superfícies, seu modelo fio-de-arame e sua representação gráfica, porém, a conversão *bottom-up* não pode ser utilizada, uma vez que as informações de um nível mais alto não podem ser obtidas das disponíveis em um nível mais baixo.

Embora a maioria dos sistemas CAD/CAM tenham incorporado a capacidade de modelagem de sólidos [Pra90], o número de aplicações que foram automatizadas é ainda pequeno, devido à concentração inicial em aspectos puramente geométricos de modelagem. Só recentemente voltou-se a atenção para requisitos adicionais de "modelos informacionalmente completos", incluindo não apenas informações geométricas, mas também outras informações associadas ao modelo, que permitam automação de processos e operação sem intervenção humana.

I.2 – CONCEITOS BÁSICOS RELACIONADOS: MODELO, MODELAGEM, CAD

Um **modelo** é um objeto construído artificialmente com a finalidade de tornar mais fácil a observação de outro objeto [Män88]. Num modelo, tenta-se abstrair apenas as informações essenciais. O nível de detalhe e informações do modelo devem ser definidos de acordo com a necessidade de sua aplicação. Um ponto chave a considerar é que muitos dos problemas para os quais se procura solução com o emprego de modelos são inerentemente geométricos. Para a resolução desses problemas com suporte computacional, a geometria do objeto é a parte mais importante do conjunto de informações que podem ser armazenadas sobre um modelo. É interessante, portanto, separarmos os dados que tratam da forma geométrica, daqueles não geométricos.

O conjunto total de dados necessários para uma classe particular de problemas é chamado **modelo do objeto**, enquanto dados puramente geométricos constituem o **modelo geométrico**. A coleção de métodos usados para definir a forma e outras características geométricas de um objeto ficou conhecida por **modelagem geométrica** a partir do início dos anos 70 [Oli91].

Métodos de modelagem geométrica são usados para construir uma descrição matemática precisa da forma de um objeto real, para projetar, analisar e visualizar formas, ou para simular algum processo. Como estas informações são usadas para uma ampla gama de finalidades, como documentação, desenho, simulação e análise em engenharia, planejamento de processos, programação e montagem automática de peças, visualização científica, entre outras, não basta a um modelo geométrico ser completo, deve ser também independente da aplicação a que se destina. O modelo deve ser criado, armazenado e analisado com o auxílio do computador. Um **sistema de modelagem geométrica** é portanto um sistema computacional que permite a criação, a modificação e o acesso à representação de objetos sólidos por meio de modelos geométricos.

A modelagem geométrica está dividida em várias ramificações, entre as quais destacam-se:

– **modelagem gráfica**: visa descrever o "desenho de um objeto", em termos de vértices e arestas, sem referenciar faces e outras características do objeto propriamente dito. Apresentam como desvantagem a geração de objetos tridimensionais incompletos e potencialmente ambíguos. Como exemplo são citados os modelos do tipo fio-de-arame.

– **modelagem de superfícies**: fornece informação detalhada sobre superfícies curvas complexas, mas nem sempre descreve um volume fechado e fornece informações suficientes para determinar todas as propriedades geométricas do objeto definido.

– **modelagem de sólidos**: enfatiza a utilização de modelos por uma vasta gama de aplicações e cria representações não ambíguas de objetos físicos sólidos, adequadas para responder a questões geométricas arbitrárias sem necessidade de interação humana.

Sistemas de modelagem fazem parte de um grande grupo de sistemas computacionais que participam do processo de projeto de produto, denominados **sistemas CAD**, que permitem manipular desenhos gerados por processos interativos entre o usuário e o sistema, pelo fornecimento de dados numéricos ou pela digitalização de imagens, fornecendo auxílio no processo de síntese, análise, otimização, avaliação e apresentação de projetos. O termo **CAD** – *Computer Aided Design*, ou Projeto Assistido por Computador, compreende a utilização de técnicas e recursos computacionais para a realização de tarefas em projetos de engenharia, arquitetura, aeronáutica, etc.

CAD pode ser visto como parte do **CAE** – *Computer Aided Engineering*, ou Engenharia Assistida por Computador, que inclui outras atividades relacionadas utilizando o computador, tais como análise de desempenho, otimização, análise de custo, planejamento de produção, especificação da necessidade de material, controle de qualidade e o controle direto da maquinaria para a manufatura do produto [Mar87]. Desta forma, atinge-se então o **CAM** – *Computer Aided Manufacturing*, ou Manufatura Assistida por Computador, que utiliza os projetos produzidos com o auxílio de sistemas CAD e os dados especificados pelo CAE para obter produtos e componentes. Este contexto mais amplo de CAD é apresentado na figura I.1: todas as informações, imagens, números e textos podem ser mantidos em bases de dados a partir das quais o usuário pode selecionar informações para criar outros desenhos, especificações e seqüências de controle numérico para obter produtos. A importância de sistemas CAD reside na produção de desenhos de melhor qualidade mais rapidamente que em tempo normal. Além disso, a reutilização de componentes de desenho proporciona maior consistência e menor redundância nos dados, programas, procedimentos de trabalho e desenhos produzidos.

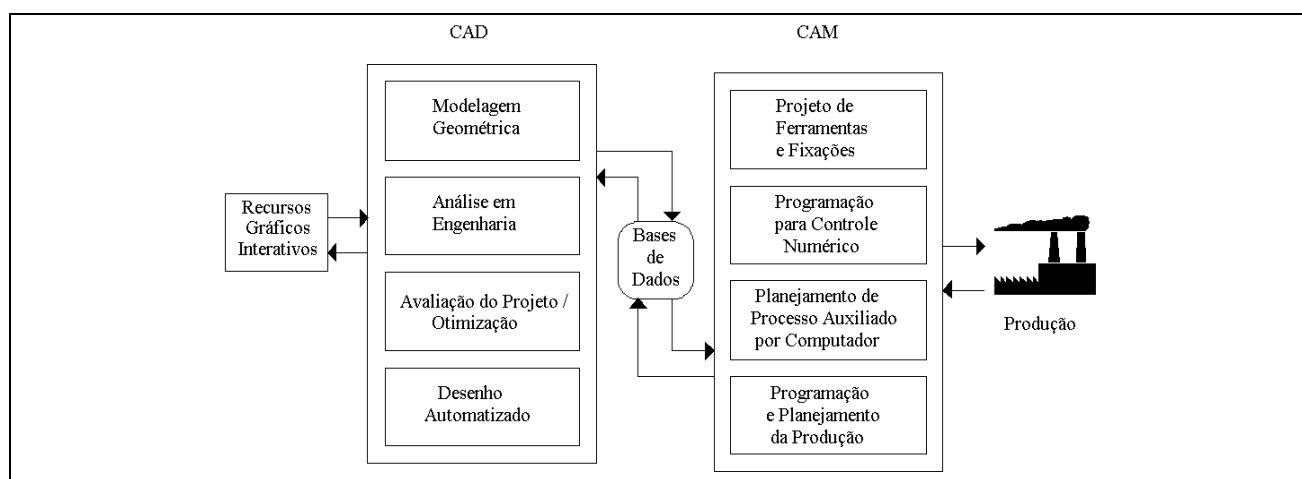



Figura I.1 – Integração CAD/CAM: da definição gráfica à produção [Mar87]

I.3 – A Modelagem Computacional de Sólidos

Segundo Requicha [Req83], o termo **modelagem de sólidos** refere-se ao conjunto de teorias, técnicas e sistemas que focalizam uma representação de sólidos "informacionalmente completa", permitindo – pelo menos a princípio – que qualquer propriedade geométrica bem definida, de qualquer sólido representável, seja automaticamente calculada. A importância dos sistemas de modelagem de sólidos está principalmente em sua capacidade de distinguir entre o interior, o exterior e a superfície de um objeto tridimensional, o que faculta calcular propriedades dependentes desta distinção [Mor85]. A fim de atingir este objetivo, algumas restrições precisam ser obedecidas, para que os sólidos modelados sejam válidos.

I.3.1 – Requisitos para a Representação Computacional de Sólidos

Um sólido é composto basicamente por faces conectadas entre si, definindo regiões. Faz-se necessária uma definição mais precisa daquilo que um modelador deve interpretar como face, uma vez que a noção intuitiva nem sempre é clara. Segundo Mortenson, as faces devem ser limitadas, orientáveis, conexas e homogêneas, devem possuir área finita, dividir o espaço em domínios disjuntos e não se auto-interceptar. Além disso, a forma a ser modelada deve conter um número finito (e maior que um) de faces que, unidas, definam a sua fronteira [Mor85]. Faces planares podem ser descritas por suas arestas limitantes, mas faces não planares requerem que as superfícies que as contêm também sejam representadas.

A capacidade de um modelador para descrever objetos que parecem sólidos não significa que ele seja adequado para representar sólidos. Nem todos os subconjuntos do espaço tridimensional correspondem a sólidos físicos válidos, ou seja, manufaturáveis. Requicha e outros pesquisadores [Req77, Req80, Mor85, Män88] estabeleceram um conjunto de propriedades necessárias para classificar um objeto como sólido abstrato representável (válido), descritas no quadro I.2.

rigidez: deve possuir uma configuração invariante, independente de sua localização e orientação no espaço.
homogeneidade tridimensional: deve possuir uma região interior definida, sem partes isoladas ou pendentes.
finitude: deve ocupar uma porção finita do espaço.
fechamento sob operações: movimentos rígidos, como translação e rotação, bem como operações de adição ou remoção de partes, quando aplicados sobre sólidos válidos, devem produzir outros sólidos válidos.
finitude de descrição: deve conter um número finito de componentes, o que permite gerar uma representação computacional do sólido.
determinismo de fronteira: a fronteira deve determinar sem ambigüidade o que está "dentro" ou "fora" do sólido.

Quadro I.2 – Propriedades necessárias para classificar um objeto como sólido abstrato válido

Apenas um pequeno grupo de subconjuntos do espaço euclidiano tridimensional (E^3) satisfazem as propriedades acima indicadas e são modelos adequados de sólidos físicos. Especialmente úteis no contexto de automação, estes subconjuntos, denominados ***r-sets***, são limitados (ocupam uma porção finita do espaço), fechados (contêm suas fronteiras), regulares (todas as suas porções possuem volumes) e semi-analíticos (podem ser expressos como uma combinação booleana finita de conjuntos analíticos, da forma $\{(x,y,z) : F_i\,(x,y,z) \leq 0\}$, sendo F_i analítica, ou seja, F_i é expansível em uma série cuja potência seja convergente sobre qualquer ponto do domínio) [Sil94, Des92]. A fronteira de um objeto separa sua parte interna de seu ambiente. Conseqüentemente, não pode haver fronteira onde não se tem interface entre objetos ou ambiente. Assim, na figura I.2, os exemplos (a) não são modelos aceitáveis para sólidos reais, mas os exemplos (b) e (c) o são. Visando garantir a manufaturabilidade, os sólidos de variedade bidimensional (*manifold*) emergiram entre os *r-sets* e se destacaram em decorrência da fundamentação teórica desenvolvida. Ilustrados na figura 1.2b, sólidos de variedade bidimensional são subconjuntos de *r-sets* caracterizados por duas propriedades principais: as arestas são compartilhadas por, exatamente, duas faces e as fronteiras são superfícies que podem ser representadas matematicamente. A figura 1.2c ilustra subconjuntos de *r-sets* que, por não apresentarem essas propriedades, recebem a denominação de sólidos de variedade múltipla (*nonmanifolds*). Esses sólidos não são manufaturáveis por serem instáveis: estão matematicamente conectados, mas sua conexão não volumétrica invalida a interpretação física como conectados; em outras palavras, um deslocamento mínimo altera sua característica de estarem conectados ou não.

a) Modelos não aceitáveis para sólidos b) sólidos de variedade bidimensional c) sólidos de variedade múltipla

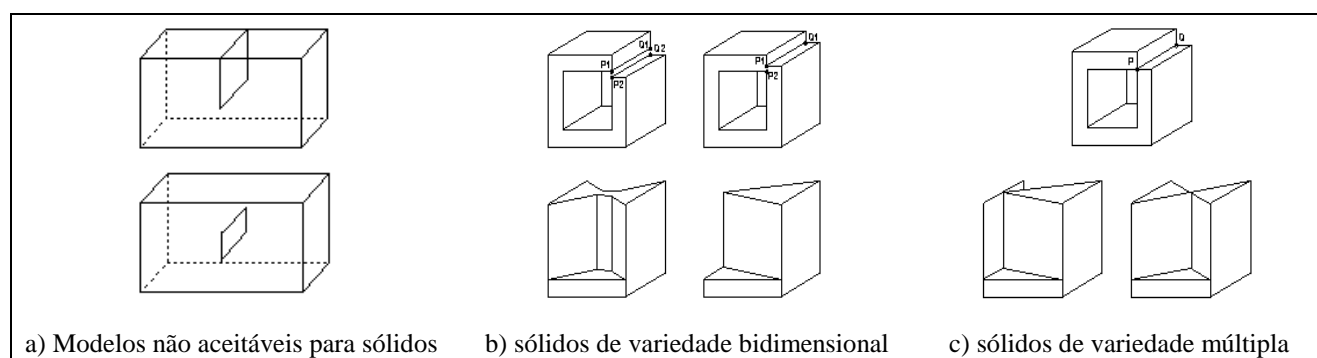
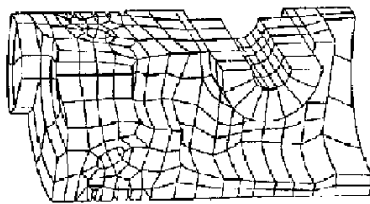
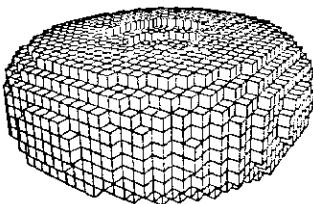
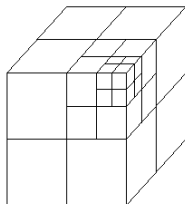
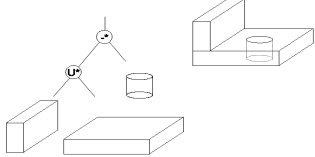
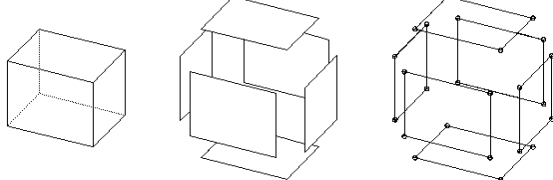
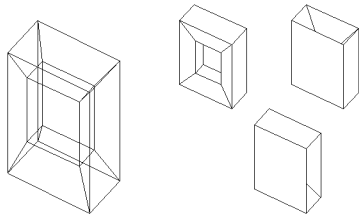
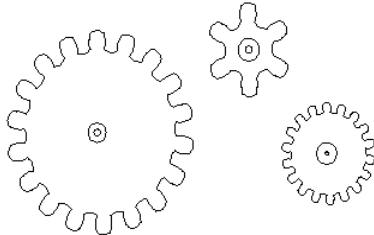
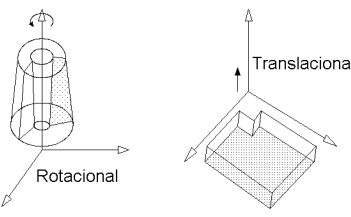



Figura I.2 – Elucidação do conceito de sólido e de variedades simples e múltipla

Um modelador de sólidos é um sistema computacional de modelagem que associa entidades geométricas (sólidos abstratos) a representações simbólicas, mediante esquemas de representação. Acompanhando a definição das propriedades para sólidos válidos, Requicha especifica uma série de propriedades desejáveis em um esquema de representação [Req80], apresentadas no quadro I.3. Algumas destas propriedades possuem significado formal para a teoria de esquemas de

representação [Gom90], enquanto outras possuem apenas significado prático, importante na implementação de um modelador específico. Vale a pena ressaltar que, diferentemente de Requicha, Mäntyla [Män88] e Foley [Fol90] incluem o fechamento sob operações como propriedade de um esquema de representação.

não ambigüidade: cada representação corresponde, sem margem de dúvida, a um único objeto;
unicidade: cada objeto modelado admite uma única representação no esquema;
potência descritiva: o conjunto de objetos passíveis de representação deve ser o mais amplo possível;
validade: qualquer representação admitida corresponde a um sólido válido (impossível criar sólido inválido);
concisão: o esquema deve ser capaz de representar sólidos armazenando o mínimo possível de informação;
facilidade descritiva: o esquema deve garantir que a criação interativa de objetos válidos seja simples;
eficiência do ponto de vista das aplicações: o esquema deve permitir o uso de algoritmos corretos, eficientes e robustos para o cálculo de funções úteis.

Quadro I.3 – Propriedades desejáveis em um esquema de representação

I.3.2 – PRINCIPAIS ESQUEMAS DE REPRESENTAÇÃO

Projetar modeladores de sólidos que satisfaçam a todas as propriedades apresentadas está longe de ser uma tarefa trivial, e geralmente não é possível abranger todas elas. A modelagem de sólidos procura definir representações que codifiquem o conjunto infinito de pontos que compõem um sólido em uma porção finita da memória do computador, da forma mais genérica possível. As representações que atendem a esta codificação podem ser divididas em três grandes grupos: **modelos de decomposição**, que representam sólidos como uma coleção de componentes básicos, combinados entre si pela operação de colagem; **modelos construtivos**, os mais utilizados no contexto atual para aplicações em CAD, os quais representam um conjunto de pontos do espaço tridimensional como uma combinação de conjuntos de pontos primitivos, incluindo operações muito mais poderosas do que a colagem, tais como as operações booleanas (união, interseção e diferença) e as transformações geométricas (rotação, translação e escalamento); **modelos por fronteira**, abordagem adotada pela maioria dos sistemas de modelagem de sólidos atualmente disponíveis, que representam sólidos como uma coleção de faces, limitadas por arestas, que por sua vez são limitadas por vértices. Existem algumas representações que também modelam sólidos, mas não se enquadram na classificação acima por possuírem características distintas. É o caso da representação fio-de-arame, que inclui todas as arestas do modelo, do Instanciamento de Primitivas, forma mais simples e direta de obter sólidos com topologia predefinida, e da representação por Varredura, forma intuitiva de especificar sólidos com simetria axial ou radial. Uma descrição sucinta dos principais esquemas de representação existentes é apresentada no quadro I.4 a seguir.

DECOMPOSIÇÃO CELULAR

- Sólido é decomposto em células, sendo que cada uma possui um número arbitrário de lados, que compartilham um ponto, uma aresta ou uma face.
- Células primitivas são parametrizadas e combinadas pela operação de colagem.
- Uso como representação básica para métodos de elementos finitos.
- Mais preciso entre os de decomposição.

[Req80, Män88, Fol90]

ENUMERAÇÃO DA OCUPAÇÃO ESPACIAL

- Sólido é visto como conjunto de cubos de tamanho fixo (*voxels - volume elements*) localizados em um *grid* espacial fixo.
- Caso especial da Decomposição Celular, no qual as células são idênticas.
- Gera arranjo ordenado de tuplas 3D, que estão ou não ocupadas pelo sólido.
- Usado em modelos onde precisão é menos importante que eficiência.

[Req80, Fol90]

OCTREES

- Divisão recursiva de uma região cúbica em oito octantes, também regiões cúbicas.
- Variação hierárquica da anterior, que visa otimizar o espaço de armazenamento.
- Gera árvore onde cada nó interno possui oito descendentes, obtidos pela subdivisão recursiva, até obter octantes homogêneos, que são as folhas com vazio / cheio.
- Ainda problemas relacionados à precisão.

[Fol90, Pra90, LaC95]

GEOMETRIA SÓLIDA CONSTRUTIVA (CSG)

- Sólidos descritos em termos de primitivas, combinadas por operadores booleanos regularizados e movimentos rígidos.
- Utilizam como representação interna árvores binárias, nas quais cada folha é uma primitiva e cada nó interno é um operador booleano (união, interseção ou diferença) ou um movimento rígido.
- Uso bastante difundido, devido ao embasamento matemático, conhecimento de algoritmos e à ampla área de aplicação.

[Req80, Män88, Hof89, Fol90, Pra90, LaC95]

REPRESENTAÇÃO POR FRONTEIRA (B-REP)

- Sólidos descritos em termos de suas superfícies limitantes.
- Topologia definida pelo conjunto de faces, arestas e vértices que compõem o sólido; geometria definida pelas coordenadas dos vértices e pelas equações das curvas e superfícies.
- A construção de sólidos válidos é garantida pelos Operadores de Euler.
- Uso amplo, devido à generalidade e disponibilidade imediata da representação de faces, arestas e das relações entre elas.

[Req80, Män88, Hof89, Fol90, Pra90, LaC95]

FIO-DE-ARAME (WIREFRAME)

- Representa um objeto por suas arestas.
- Consiste inteiramente de pontos, linhas e curvas, dando ilusão de solidez.
- A falta de informação geométrica ao modelar sólidos resulta em problemas: geração de modelos ambíguos ou sem sentido físico; falta de coerência visual; confusão devido ao excesso de linhas.
- Eficiente para a geração e representação de desenhos.

[Voe77, Req80, Mor85, Män88, Pra90]

INSTANCIAMENTO DE PRIMITIVAS

- Define um conjunto de formatos sólidos primitivos que são relevantes para a área em que os modelos serão utilizados.
- Primitiva define uma família de componentes cujos membros variam em relação a alguns parâmetros.
- Objetos representados por tuplas de tamanho fixo (ex.: "esfera", centro, raio).
- Uso combinado com outros esquemas para definir objetos complexos.

[Req80, Män88, Fol90]

VARREDURA (SWEEP)

- Representação baseada na noção de mover uma região (gerador) por um caminho (diretor), utilizando duas formas: rotacional ou translacional, que podem ser combinadas entre si e com o escalamento.
- Uso em automação industrial e animação
- Utilizada por vários esquemas como alternativa para a descrição de objetos.
- Forma natural e intuitiva de construir uma grande variedade de objetos.

[Req80, Shi83, Män88, Fol90, Cas90]

Quadro I.4 – Principais esquemas de representação de sólidos

Do ponto de vista da construção de modelos, nenhuma das representações possui propriedades uniformemente melhores que as outras. Para conseguir o máximo de flexibilidade em um sistema de modelagem de sólidos, representações híbridas [Mil89, Gom90, Fis91] são utilizadas para combinar esquemas diferentes, procurando explorar o potencial de cada representação. Do ponto de vista da edição de modelos, entretanto, a situação é diferente. Existem técnicas de edição poderosas, baseadas na geometria construtiva, que não podem ser implementadas de maneira razoável em um esquema por fronteiras, uma vez que este não mantém um registro do processo de construção. A habilidade de modificar, substituir ou remover primitivas ou sub-árvores numa árvore da geometria construtiva permite que modificações significativas num modelo sejam especificadas com um mínimo de comandos por parte do projetista. Da mesma forma, existem métodos poderosos baseados em representação por fronteiras que não podem ser implementados num contexto puramente construtivo.

Uma vez que tanto a modelagem por fronteira quanto a construtiva fornecem representações de sólidos não ambíguas, qualquer consulta que dependa unicamente da geometria e atributos a ela relacionados pode, a princípio, ser respondida por qualquer uma delas. Diante desta constatação, Miller [Mil89] apresentou um sistema de modelagem ideal, ilustrado na figura I.3, capaz de manter e manipular ambas as representações de maneira redundante, explorando as vantagens de cada uma. Infelizmente, a estrutura híbrida de dados é muito mais complexa, tornando ainda mais difícil computar as propriedades dos sólidos resultantes. Para aplicações mais comuns, algoritmos que operam em mais de uma representação têm sido desenvolvidos, entretanto, certas classes de algoritmos são mais adequadas para uma representação do que para outra, em termos de desempenho e confiabilidade. Quanto à conversão entre modelos, existem significativos problemas arquiteturais, conceituais e práticos, que precisam ser resolvidos *a priori*, para que seja possível realizá-la [Tor84, Dob88, Sha91].

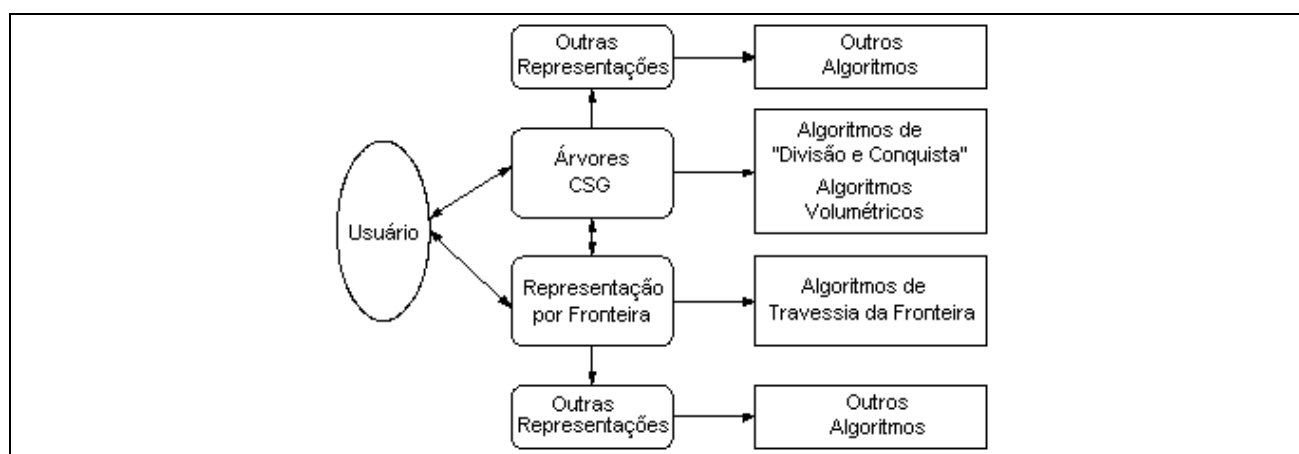



Figura I.3 – Arquitetura híbrida ideal [Mil89]

I.4 – O MÉTODO DE ELEMENTOS FINITOS E A MODELAGEM DE SÓLIDOS

Originário das técnicas de cálculo de estruturas mecânicas, o método de elementos finitos é uma ferramenta que simula normalmente um fenômeno ou um processo físico por meio de um conjunto de equações envolvendo derivadas parciais e de condições de contorno a elas associadas. Devido à flexibilidade, à facilidade de programação e à ligação direta com a física dos fenômenos estudados, o método de elementos finitos é, no domínio da física matemática, o método numérico universalmente mais utilizado para cálculo de campos [Sab93].

O princípio fundamental do método de elementos finitos reside na discretização do domínio de estudo em domínios elementares de dimensão finita, denominados elementos finitos. A cada um destes elementos é associada uma função, geralmente aproximada por um polinômio cujo grau pode variar de uma aplicação para outra. Um elemento é caracterizado pelo número de nós geométricos e pelo grau de aproximação da função procurada em seu domínio, podendo possuir lados retilíneos ou curvos. A topologia mais simples destes elementos é triangular (2D) ou tetraédrica (3D), sendo possível a utilização de quadriláteros (2D) ou poliedros (3D). Elementos disjuntos, triangulares ou quadriláteros, retilíneos ou curvilíneos, devem realizar uma partição do domínio de estudo de forma que a sua união abranja todo o domínio. Geralmente denominada de discretização do domínio, esta partição deve respeitar regras que assegurem um bom desenvolvimento do cálculo.

I.4.1 – ARQUITETURA GERAL DE SISTEMAS BASEADOS NO MÉTODO DE ELEMENTOS FINITOS

A simulação de um problema, normalmente, compreende três etapas, mostradas na figura I.4: o pré-processamento, que realiza a descrição completa do problema, descrevendo a geometria, realizando a discretização do domínio e definindo características físicas e condições de contorno; o processamento, que envolve a resolução pelo método de elementos finitos, expressando o problema sob a forma de um conjunto de equações que são em seguida resolvidas; o pós-processamento, que transforma as informações obtidas na etapa anterior em grandezas compreensíveis pelo usuário, permitindo a visualização, interpretação e exploração dos resultados da simulação.

Essas três etapas são geralmente bem distintas, constituindo três módulos independentes em um software: a entrada, o cálculo e a saída. Numa configuração mínima, os módulos de entrada e de saída podem estar ausentes, sendo os dados introduzidos por um editor de texto e os resultados

a) funções do pré-processador b) funções do processador c) funções do pós-processador

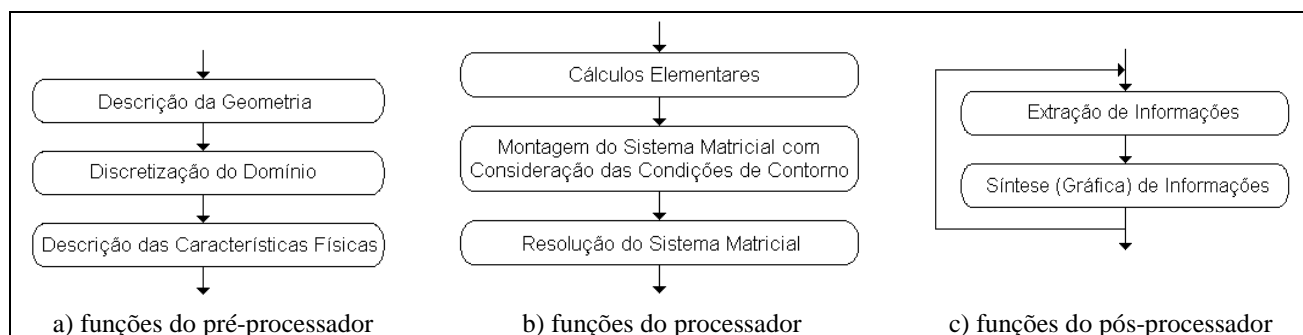



Figura I.4 – Tarefas executadas pelo pré-processador, processador e pós-processador

procurados entre a totalidade dos resultados de um arquivo. Esta forma de trabalho exige um esforço significativo do usuário e em muitos casos induz ao erro. A fim de reduzir o tempo de aquisição de dados e de avaliação dos resultados da simulação, os módulos de entrada e de saída encontram-se geralmente desenvolvidos, exigindo uma interatividade alfanumérica e gráfica de alto desempenho. Por outro lado, o módulo de cálculo exige principalmente os recursos clássicos de cálculo científico: operações aritméticas, memória principal e memória de massa. Essa heterogeneidade de requisitos quanto a recursos computacionais leva a variações na organização dos softwares existentes [Sab93], como as descritas no quadro I.5. A separação entre os módulos pressupõe que exista alguma forma de intercâmbio de dados entre eles. Como estes módulos geralmente constituem programas concebidos independentemente uns dos outros, torna-se necessário realizar “traduções” entre eles. A fim de limitar o número de ”traduções”, é preferível definir uma base de dados com formato padrão com a qual todos os módulos se comunicam, denominada neste trabalho de "arquivo neutro”.

a) todos os módulos agrupados	b) entrada e saída agrupados	c) todos os módulos separados
Programa único gerenciando todas as funções, permitindo uma passagem rápida entre as diferentes operações, sendo particularmente adaptado às aplicações que necessitam de numerosos ciclos de entrada-cálculo-saída.	O usuário pode escrever seu problema e explorar os resultados em um meio interativo de alto desempenho e efetuar os cálculos em *batch* ou mesmo *on line* em um computador realmente potente (ex.: vetorial)	Construção de uma cadeia de tratamento por elementos finitos, podendo existir malhas complementares (eventualmente de origens diferentes) e programas encadeados formando uma seqüência de resolução.

Quadro I.5 – Organizações encontradas em softwares de elementos finitos

I.4.2 – A DESCRIÇÃO DA GEOMETRIA POR MEIO DE UM MODELADOR DE SÓLIDOS

Entre os esquemas para a descrição da geometria do objeto pelo módulo de entrada, destacam-se as representações CSG e B-rep, detalhadas nos capítulos IV e V, respectivamente. Do ponto de vista do usuário, as informações que descrevem a geometria poderão ser introduzidas por um arquivo contendo as informações codificadas manualmente, por um editor de texto, ou por um software de CAD que possua uma interface com o software de simulação. Cada método possui suas vantagens e inconvenientes. A geração manual de um arquivo utilizando um editor de textos padrão necessita muito mais cuidado por parte do usuário para descrever o problema sem erro, sendo necessário respeitar um formato relativamente rígido. Este método justifica-se em uma utilização pouco freqüente de um programa ou durante o desenvolvimento do software. Para uma utilização intensa, é preferível trabalhar com um módulo de entrada interativo, que permita o fornecimento e o controle gráfico dos dados. Desta forma, a descrição de problemas torna-se mais fácil, mais rápida e sobretudo mais confiável.

A presença de um único software, para cumprir as funções de projeto e de entrada de dados para a simulação, simplifica o treinamento e o gerenciamento de projetos, uma vez que as informações necessárias encontram-se em uma só base de dados. A utilização de um modelador de sólidos é particularmente útil para descrever, verificar e visualizar a geometria de objetos tridimensionais, porém os algoritmos para a geração de malhas a partir de uma geometria tridimensional são ainda pouco padronizados, principalmente porque a maior parte das malhas tridimensionais são obtidas com a ajuda de métodos específicos, sem se prender a um modelador de sólidos. Outro problema é que uma única descrição computacional para um objeto pode não satisfazer a todas as suas aplicações. Visando diminuir a complexidade da malha resultante, muitas vezes é necessário simplificar a geometria do objeto, porém mantendo-a ainda aceitável. A obtenção automática da geometria simplificada a partir de uma geometria completa é uma técnica que requer alguns cuidados [Arm94].

I.4.3 – TÉCNICAS DE DISCRETIZAÇÃO DO DOMÍNIO

A discretização do domínio corresponde à passagem do meio contínuo à sua representação discretizada. Para o método de elementos finitos, a discretização resulta da divisão do domínio em um conjunto de sub-domínios – os elementos – respeitando as fronteiras e as interfaces do domínio

inicial. Esta discretização implica o aparecimento de um certo número de nós onde serão definidos os graus de liberdade utilizados nas equações dos elementos finitos. A utilização de elementos simples (triângulo e quadrilátero em 2D; tetraedro, prisma e hexaedro em 3D), eventualmente curvilíneos, permite dividir perfeitamente todo objeto bi ou tridimensional. A relativa facilidade de discretização em elementos finitos, assim como a grande generalidade dos procedimentos numéricos associados, fazem com que o método de elementos finitos seja largamente utilizado nas ferramentas de CAE [Sab93].

Como o método de elementos finitos pressupõe que as funções de forma sejam contínuas, é fundamental manter a continuidade entre os elementos durante a discretização. A continuidade implica a existência de uma compatibilidade entre as discretizações do domínio e da fronteira, conforme ilustrado na figura I.5. A interpretação de maneira única da fronteira entre objetos força à compatibilidade entre os vértices da fronteira e os nós da malha superficial, ou seja, os nós que estão sobre a aresta comum a duas faces da fronteira do objeto devem pertencer a estas duas faces e conter os vértices definidos sobre esta aresta. Para um elemento finito de fronteira situado numa interface entre regiões existem dois elementos finitos vizinhos: um à direita e outro à esquerda de um elemento unidimensional em 2D e um acima e outro abaixo de um elemento bidimensional em 3D. Para um elemento finito de fronteira situado nas bordas do domínio, existe apenas um único elemento vizinho.

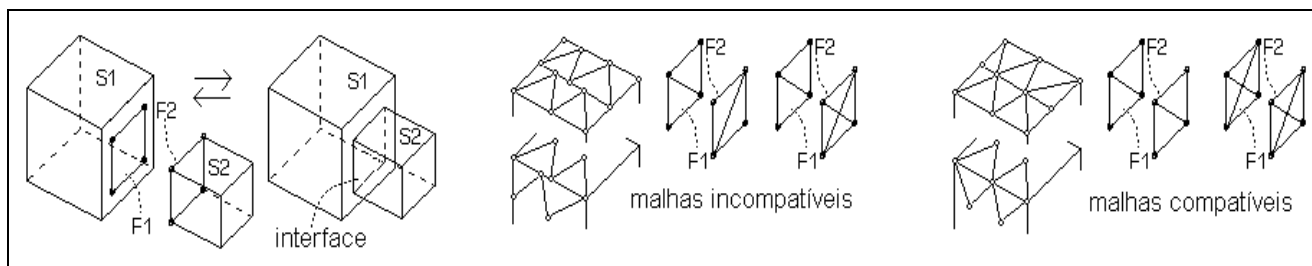
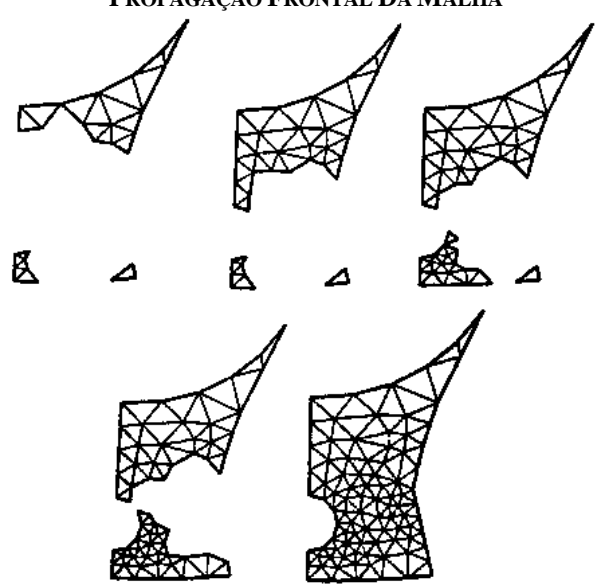
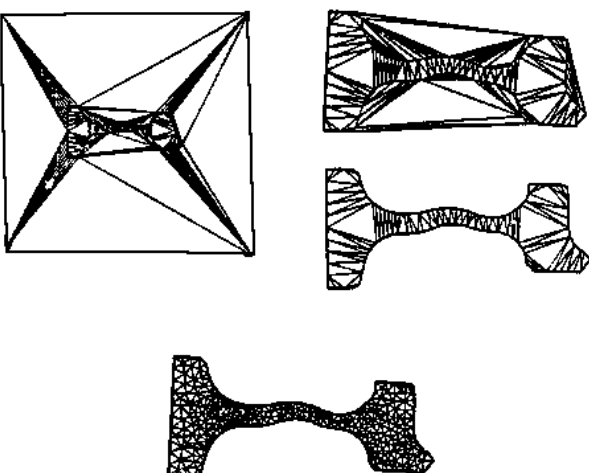
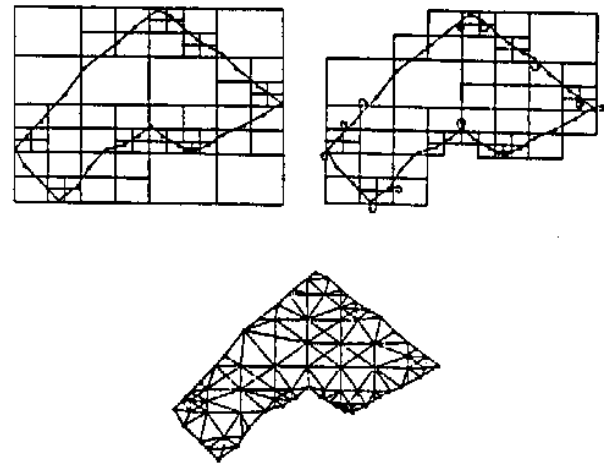
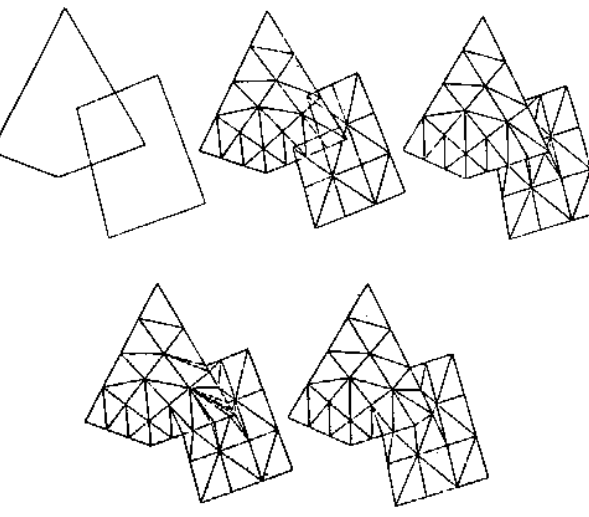



Figura I.5 – Elucidação do conceito de compatibilidade na discretização

Todas essas informações, dadas desta forma ou de outra equivalente, são necessárias para um tratamento por elementos finitos. Entretanto, elas são excessivamente enfadonhas se codificadas manualmente. Por esta razão, vários métodos de geração automática de malha foram criados [Geo91], e os de maior interesse para este trabalho são apresentados no quadro I.6. As ilustrações apresentadas são bidimensionais apenas para facilitar a compreensão dos métodos, pois o mesmo raciocínio é válido tanto para 2D como para 3D.

Propagação Frontal da Malha

– Discretiza o volume a partir da fronteira discretizada em elementos superficiais, utilizando tetraedros.
– Processo iterativo, no qual os elementos são construídos camada por camada, apoiando-se sobre a frente de propagação.
– Etapas: análise da região; definição de pontos internos; criação de tetraedros pela união desses pontos às faces, obtendo nova região a ser analisada; reinício do processo,até obter região vazia
– Primeira solução automática para a geração de malhas de domínios com fronteiras arbitrárias.
– Dados fornecidos pelo modelador: fronteiras do objeto discretizadas em elementos superficiais. [Geo91]

Método de Voronoï - Delaunay

– Processo: definição de um hexaedro que englobe o domínio; criação da malha do hexaedro contendo 5 tetraedros; malhamento dos tetraedros sem a inserção de novos pontos, utilizando processo específico; eliminação dos tetraedros exteriores ao domínio; refinamentos e suavização da malha final.
– Triangulação dependente do número e localização dos pontos que discretizam o contorno do problema e do número e localização dos pontos internos.
– Dados fornecidos pelo modelador: fronteira do objeto discretizada em elementos superficiais. O gerador de malha deve criar pontos internos adicionais. [Geo91]

Decomposição Espacial Recursiva

– Partindo de uma região cúbica envolvendo o domínio, uma divisão recursiva do espaço em octantes é processada toda vez que este estiver parcialmente ocupado pelo objeto, até que o octante esteja totalmente dentro ou fora do objeto.
– Neste ponto, os octantes existentes na superfície são decompostos em combinações de tetraedros e pirâmides, procurando melhorar a representação da fronteira do objeto.
– Dados fornecidos pelo modelador: árvore de partição de espaço gerada pela representação *Octree* referente à divisão recursiva em octantes. [Geo91]

CSG com Malhamento Sobre Primitivas

– Primitivas já discretizadas (por qualquer outro método) são combinadas entre si utilizando as operações booleanas de união, interseção e diferença, e uma nova malha é obtida com base na malha inicial gerada sobre essas primitivas.
– Distingue-se dos outros métodos por gerar a malha sobre as primitivas, mantendo em paralelo a representação CSG e a representação discretizada.
– Dados fornecidos pelo modelador: árvore binária gerada pela representação CSG, contendo primitivas em nós folhas e operadores booleanos em nós internos. [Kam91]

Quadro I.6 – Principais métodos de geração automática de malha

A qualidade da malha está intimamente relacionada à sua forma e ao seu tamanho: os elementos não devem afastar-se das formas ideais (triângulos equiláteros, quadrados, tetraedros equiláteros, cubos, etc.), sob pena de degradar a solução. Já a dimensão dos elementos influi no erro do método introduzido pela discretização, sendo necessário diminuir o tamanho e aumentar a concentração de elementos onde a solução apresenta-se mais perturbada.

Visando melhorar a qualidade da malha, muitas vezes torna-se necessário aplicar algum processo de refinamento. De uma maneira geral, existem dois tipos de refinamentos: o tratamento das formas dos elementos – obtida deslocando cada nó interno do polígono para o seu baricentro, como ilustrado na figura I.6 [Sab93] – e a otimização da malha – obtida pela redução do tamanho dos elementos ou pelo aumento da precisão sobre o elemento. Nos geradores de malha adaptativos, a malha e a resolução estão estreitamente ligadas. A malha ótima depende não só da geometria, mas também do problema físico.

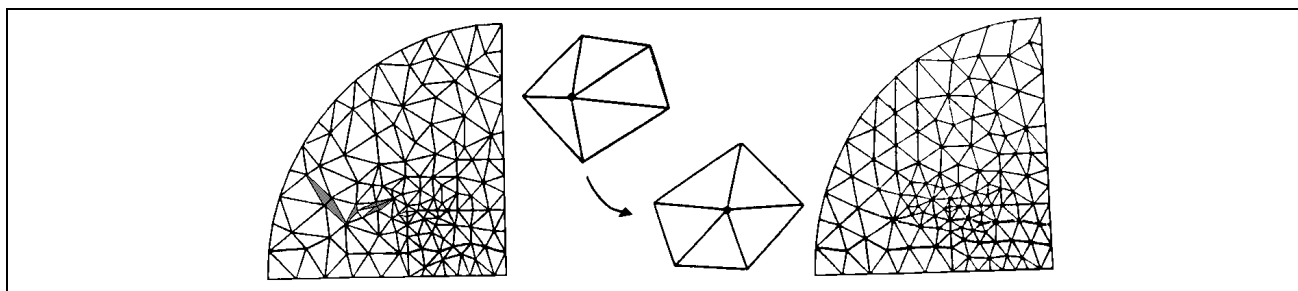



Figura I.6 – Regularização de uma triangulação por meio do deslocamento do nó central para o baricentro do polígono

I.4.4 – A Atribuição de Características Físicas

Além da definição geométrica do domínio e de sua discretização, o modelamento numérico necessita do tipo da equação diferencial a ser resolvida e das características referentes aos coeficientes dessas equações. É também necessário obter o tipo e as características das condições de fronteira e de seus coeficientes. A definição das características físicas corresponde à identificação das diferentes partes do domínio – as regiões – e das diferentes partes das bordas do domínio – as fronteiras – que constituem o modelo e intervêm no processo.

A especificação das características físicas consiste em descrever efetivamente as características dos materiais e das fontes de cada uma das regiões, bem como o tipo e os parâmetros das condições de contorno em cada uma das fronteiras. Essas informações podem ser fornecidas fora da descrição geométrica e do gerador de malhas sendo, portanto, genéricas e independentes da simulação. Nesse caso, a ligação entre a topologia e as características físicas é feita por intermédio de nomes associados às regiões e às fronteiras durante a fase de descrição topológica.

I.5 – MOTIVAÇÕES E OBJETIVOS

Na busca incessante das indústrias por uma maior produtividade, o caminho encontrado tem sido a automação da manufatura integrada pelo computador, ou CIM. Fatores como velocidade, custo e flexibilidade têm sido decisivos na opção por modelos computacionais ao invés de protótipos reais. Desde os primeiros esboços até a construção detalhada do produto, a modelagem de sólidos integra o CIM, propiciando a automação não só do processo de desenho, mas também da verificação do projeto, abrindo caminho para processos automatizados de simulação, análise e otimização. Mais do que isso, a modelagem de sólidos permite a integração entre as etapas de projeto e manufatura, uma vez que viabiliza a comunicação inequívoca dos resultados do projeto para a produção.

Devido, principalmente, ao seu potencial de automatização de aplicações, a modelagem de sólidos tem evoluído bastante. O tempo de latência – tempo requerido para transformar o resultado de uma pesquisa em um produto de mercado – é menor na modelagem de sólidos do que em muitas outras tecnologias CAD/CAM [Req83, Req92], porém esse fator não está sendo suficientemente explorado na área de engenharia elétrica. Boa parte dos modeladores voltados para aplicações em elementos finitos definem modelos 3D apenas justapondo objetos definidos por varredura de superfícies 2D, não explorando outras técnicas poderosas existentes, como as representações CSG e B-rep. Como exemplo, a construção de modelos complexos, em especial os assimétricos, é normalmente muito mais simples e intuitiva utilizando a composição de primitivas do que pela definição de inúmeras superfícies e curvas varridas pelo espaço. Além disso, um menor número de componentes permite agilizar a construção de um modelo e as informações geométricas e topológicas disponíveis possibilitam verificar automaticamente sua integridade e manufaturabilidade.

Mais importante do que o emprego de outros esquemas de representação, existe uma lacuna entre poder descritivo e manufaturabilidade que tem dificultado o tratamento adequado de fronteiras entre regiões, também denominadas fronteiras internas. A noção de *r-sets*, apresentada no item I.3.1, exclui uma classe de objetos de grande interesse para o eletromagnetismo: os que admitem fronteiras internas para representar a interface entre regiões, necessárias quando existem partes de um objeto compostas por diferentes materiais em contato direto, além de regiões de ar que envolvem estes objetos e podem conter campos eletromagnéticos.

Para o eletromagnetismo é necessário um modelador capaz de definir componentes sólidos manufaturáveis que possam ser acoplados entre si, gerando fronteiras internas unicamente representadas, capazes de garantir a compatibilidade da malha de elementos finitos gerada. Se cada

região fosse tratada de maneira independente, sem compartilhar faces e arestas comuns, malhas superficiais distintas e incompatíveis poderiam resultar em cada região, impossibilitando seu correto tratamento pelo método de elementos finitos.

Uma solução apresentada por vários modeladores de sólidos atuais consiste no acoplamento de faces e arestas ao modelo, gerando objetos de variedade não bidimensional (*nonmanifold*). Esta solução permite não só a inclusão de uma face bidimensional dividindo uma região em duas – o que corresponderia a uma fronteira interna – mas também a inclusão de quaisquer faces e arestas em todo o modelo, destruindo a homogeneidade tridimensional bem como o determinismo da fronteira e possibilitando a geração de modelos fisicamente não realizáveis (não manufaturáveis). Algoritmos para verificar e garantir a manufaturabilidade dos componentes modelados acabam sendo complexos e ineficientes. A melhor solução, portanto, é possibilitar a inclusão de fronteiras internas em sólidos de variedade bidimensional e garantir que as operações aplicadas sobre eles também produzam sólidos de variedade bidimensional válidos.

Procurando resolver o problema de conciliar a manufaturabilidade com a presença de fronteiras internas, esta tese tem por objetivo submeter à discussão e à crítica a construção de um sistema CAD voltado para aplicações em eletromagnetismo que atenda aos seguintes requisitos:

– permita construir modelos CSG combinando primitivas instanciadas ou definidas por varredura, o que é intuitivo e de fácil utilização pelo usuário;

– derive do modelo CSG a representação B-rep, obtendo vértices, arestas e faces do modelo, que facilitem a geração da malha superficial;

– possibilite a definição, fundamental para o eletromagnetismo, de sólidos manufaturáveis contendo fronteiras internas;

– gere uma malha superficial poligonal de boa qualidade, a ser usada como ponto de partida para a obtenção da malha volumétrica em um gerador automático de malhas adaptativas que utiliza o método de Delaunay;

– seja capaz de fornecer as informações necessárias para a simulação eletromagnética do modelo, incluindo não só a descrição geométrica, mas também os nós, as arestas e os elementos superficiais que compõem a malha, além de detalhes sobre o cálculo, condições de contorno e propriedades físicas dos materiais envolvidos no problema;

– permita o intercâmbio de dados, quer seja com outras etapas do processamento, ou com outros projetos desenvolvidos pelo grupo de pesquisa, pela geração de uma base de dados contendo o formato neutro definido pelo GOPAC [Roc95, Gop96];

– possibilite a parametrização de sua geometria, para ser futuramente incluído em um sistema de otimização da forma de dispositivos eletromagnéticos, no qual será possível manipular uma família de objetos com a mesma topologia e executar ciclos sucessivos envolvendo adequação de parâmetros, simulação e análise, visando obter um projeto otimizado do produto a ser construído.

A fim de facilitar a manutenção e a evolução do sistema, procurou-se ainda selecionar técnicas adequadas de Engenharia de Software e utilizá-las durante o desenvolvimento, de forma a produzir um sistema bem documentado, modular, que possa ser estendido, aprimorado e, principalmente, integrado a outros módulos de software.

I.6 – ORGANIZAÇÃO DO TRABALHO

Este trabalho encontra-se dividido em oito capítulos.

O primeiro, Introdução, inicia-se com um breve histórico da evolução da modelagem, acompanhado de alguns conceitos básicos necessários à compreensão e desenvolvimento do tema. Em prosseguimento, foram descritos conceitos, propriedades, técnicas e características comuns aos modeladores de sólidos, fornecendo uma fundamentação teórica sobre sua modelagem computacional. Contextualizando, foram apresentadas algumas características do método de elementos finitos e seu envolvimento com modeladores de sólidos. Em consideração a necessidades específicas da área de eletromagnetismo, foram relacionadas as bases em que se apoiou a proposta deste trabalho e alguns requisitos para o tratamento do problema.

O Capítulo II relaciona os principais requisitos do modelador em construção pelo GOPAC/UFMG e define claramente o escopo desta tese em relação ao projeto global. A estrutura funcional do modelador é então apresentada, identificando os Subsistema de Interface, Modelagem, Representação e Geração de Malha. A seguir, são descritas as características mais importantes relacionadas a projeto, implementação e documentação.

Dedicado ao Subsistema de Interface, o Capítulo III detalha o ambiente e a interação com o usuário, mostrando os recursos disponíveis, como é realizado o tratamento de comandos e a entrada de dados, como se manipula a área de trabalho, as formas e os recursos utilizados para a visualização de modelos, bem como as informações que podem ser extraídas do modelo e armazenadas pelo modelador.

O Subsistema de Modelagem é detalhado no Capítulo IV, que desenvolve um estudo mais aprofundado sobre a estrutura CSG, abrangendo a definição de primitivas e elementos geométricos, a utilização das operações booleanas e de montagem, bem como das transformações geométricas, que podem também ser aplicadas fora do contexto CSG.

O Capítulo V descreve as principais estruturas de dados gerenciadas pelo Subsistema de Representação ou Núcleo, destacando a representação B-rep proposta e construída com o emprego dos operadores de Euler definidos, o que constitui a principal inovação deste trabalho. A necessidade, as etapas e as dificuldades envolvidas na conversão entre a representações CSG e B-rep também são abordadas nesse capítulo.

No Capítulo VI são apresentados os principais aspectos relacionados ao Subsistema de Geração de Malha, mostrando como pode ser obtida uma malha superficial homogênea e de boa qualidade diretamente sobre primitivas definidas por varredura, além de como funciona a atribuição de características físicas e dados de cálculo aos componentes do modelo.

A fim de ilustrar o trabalho desenvolvido e o potencial descritivo do modelador, o Capítulo VII mostra alguns modelos gerados para problemas eletromagnéticos. Também apresenta conceitos e detalhes sobre a geração, pelo modelador, da base de dados neutra desenvolvida pelo GOPAC.

O Capítulo VIII discute alguns pontos relevantes, destaca as contribuições deste trabalho e sugere aperfeiçoamentos a serem desenvolvidos em futuros trabalhos relacionados.

O Apêndice I apresenta a padronização de código e documentação adotada para módulos e programas durante o desenvolvimento do modelador, enquanto o Apêndice II relaciona informações básicas sobre os comandos disponíveis no sistema.

Finalizando, é apresentada a bibliografia consultada, tanto para o referencial teórico quanto para o metodológico.

"Quando sonhamos sozinho, é apenas um sonho.
Quando sonhamos juntos, é o começo da realidade."
D. Helder Câmara

II – DESENVOLVIMENTO DE UM MODELADOR DE SÓLIDOS VOLTADO PARA APLICAÇÕES EM ELETROMAGNETISMO

De maneira geral, os sistemas de modelagem de sólidos possuem componentes básicos, tais como uma estrutura de dados para armazenar os esquemas de representação, um conjunto básico de algoritmos para a manipulação dessas estruturas e mecanismos de interface com o usuário que permitam implementar diferentes técnicas de modelagem e edição. Mas isto não é tudo. Baseado no estudo de requisitos e funcionalidade para modeladores de sólidos em geral, nas necessidades específicas levantadas para as áreas de eletromagnetismo e elementos finitos [She85, Mag98a], e em regras de engenharia de software, este capítulo descreve as características gerais de projeto, implementação e documentação utilizadas no modelador de sólidos em construção pelo GOPAC / CPDEE.

A seção II.1 faz um breve histórico do Projeto Integrado de Pesquisa e a seção II.2 define claramente o escopo desta tese em relação a ele. Dando continuidade, a seção II.3 destaca as mais importantes características de projeto do modelador, entre elas as especificações funcional e de requisitos. A seção II.4 descreve características gerais de implementação, indicando as regras adotadas para garantir uma boa modularização para o sistema. Visando à manutenção e à evolução do modelador, a seção II.5 apresenta o padrão de documentação definido para o sistema.

II.1 – UM BREVE HISTÓRICO

Encontra-se em andamento no Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica da Universidade Federal de Minas Gerais – CPDEE/UFMG – um Projeto Integrado de Pesquisa do GOPAC – Grupo de Otimização e Projeto Assistido por Computador, voltado para o desenvolvimento de pesquisas em eletromagnetismo que utilizam o método de elementos finitos, com o objetivo de aprofundar conhecimentos e implementar métodos para geração adaptativa de malha de elementos finitos em três dimensões usando o método de Delaunay.

No início do projeto, foi desenvolvido um gerador automático de malhas tridimensionais que utilizava dados gerados pelo programa AutoCAD, obtidos com o módulo opcional denominado AME (*Advanced Modeling Extension*) [Aut92]. Infelizmente, a malha superficial assim obtida incluía triângulos de baixa qualidade. Procurou-se então recorrer ao módulo de programação ADS (*AutoCAD*

Development System) [Aut92a] que permite a modificação das funções existentes ou o desenvolvimento de novas funções no AutoCAD, utilizando a linguagem C, porém mesmo usando o módulo de programação ADS, graves restrições puderam ser notadas no programa. A principal delas refere-se à limitação a algumas geometrias específicas, incluindo somente objetos de forma bastante simples, inviabilizando sua aplicação a objetos reais que incluem, normalmente, formas geométricas bastante complexas, as quais podem ser derivadas de formas mais simples. Esta limitação está associada ao sistema de geração de malhas tridimensionais disponível no programa.

Na impossibilidade de resolver o problema utilizando o AutoCAD, tornou-se necessário o desenvolvimento de uma nova ferramenta para a geração da malha de elementos finitos. Esta ferramenta seria desmembrada em duas partes, a serem desenvolvidas em dois trabalhos de doutorado. A primeira delas, tema deste trabalho, refere-se ao desenvolvimento de um modelador de sólidos capaz de construir modelos manufaturáveis combinando primitivas CSG instanciadas ou definidas por varredura, derivando a representação por fronteira (B-rep) a partir da representação CSG e gerando a malha de elementos finitos superficial sobre esta fronteira. A segunda parte utiliza como ponto de partida a malha superficial e volumes obtidos na primeira e, realizando a tetraedrização dos volumes pelo método de Delaunay e refinamentos necessários, gera malhas volumétricas auto-adaptativas [Gon98, Gon00].

II.2 – Escopo deste Trabalho

A proposta básica deste trabalho consiste no desenvolvimento do modelador de sólidos cuja especificação funcional e de requisitos, apresentada a seguir, considerou o projeto como um todo, definindo um modelador que supre, a princípio, todas as necessidades levantadas para a área de eletromagnetismo. É importante, porém, ressaltar que nem todos os requisitos especificados serão aqui abordados, uma vez que é impossível satisfazer a todos eles em um único trabalho de doutoramento. O modelador proposto resultará de um esforço conjunto, envolvendo outros trabalhos a este relacionados. Assim sendo, é importante definir claramente o escopo aqui pretendido, que abrange:

– a implementação da interface com o usuário, incluindo tratamento de comandos e passagem de parâmetros, mudança de plano de trabalho e recursos valiosos de visualização;

– a implementação das rotinas referentes a geração e manipulação de primitivas pelas transformações geométricas, bem como a definição de operações booleanas e a construção de árvores CSG;

– a implementação parcial da avaliação da fronteira, incluindo a triangularização de primitivas e a realização de casos mais simples de operações booleanas e de montagem sobre modelos poliedrais, obtendo como resultado uma representação B-rep e uma malha superficial aproximadas, em condições de incorporar por completo as operações booleanas e, futuramente, curvas e superfícies mais elaboradas;

– a construção das rotinas necessárias para manipulação e armazenamento das estruturas de dados presentes no núcleo do modelador, bem como das rotinas de leitura e gravação de arquivos de dados envolvidos;

– a geração da malha superficial do modelo a partir da triangularização produzida sobre as primitivas presentes na árvore CSG;

– a implementação de um mecanismo para a definição e atribuição, aos componentes do modelo e por meio de *labels*, das propriedades físicas e dos dados relacionados ao processamento;

– a construção e armazenamento de uma base de dados com formato neutro, contendo a descrição da geometria e da malha, bem como a definição de *labels* e materiais empregados.

II.3 – Características Gerais de Projeto

II.3.1 – Especificação Funcional

O modelador de sólidos em desenvolvimento pelo GOPAC deve ser capaz de fornecer uma descrição completa do modelo, incluindo geometria, fronteiras entre regiões, características físicas e outros atributos específicos para o eletromagnetismo. Deve suportar vários tipos de primitivas tridimensionais, permitindo a criação, edição e acesso à representação de objetos sólidos por meio de processos interativos ativados pelo usuário. Para que isto seja possível, rotinas de visualização devem ser incorporadas, de forma a permitir uma melhor observação da configuração geométrica do modelo em construção. Visando à simulação eletromagnética pelo método de elementos finitos, rotinas para a geração automática da malha superficial deverão ser acopladas, para que programas aplicativos externos possam utilizar as informações disponíveis para analisar o modelo de forma estática ou dinâmica, determinando as distribuições de campos elétricos e magnéticos e, a partir destas, grandezas como indutâncias, forças, torques, resistências, perdas, capacitâncias, etc.

O sistema deverá suportar a representação parametrizada das primitivas sólidas bloco, prisma, pirâmide, cilindro, cone, esfera, elipsóide e toro. Com o objetivo de aumentar o potencial descritivo do modelador, esse conjunto de primitivas deverá ser ampliado de modo a incluir primitivas planares, sobre as quais serão realizadas operações de varredura translacional e rotacional.

Grosso modo, o modelador projetado deverá ser funcionalmente composto por quatro subsistemas principais, ilustrados na figura II.1, os quais devem ser desenvolvidos e operar de forma independente, ainda que colaborem entre si e estejam funcionalmente interligados:

– o Subsistema de Interface, responsável por propiciar ao usuário as diversas maneiras de descrever objetos sólidos, bem como um conjunto de operações de manipulação da forma que possam ser executadas sobre esses objetos. Deve também fornecer recursos para a visualização dos objetos representados e interfaces para comunicação com outros modeladores e aplicativos. As tarefas a serem realizadas pelo modelador são descritas e captadas por procedimentos da Interface, que ativam procedimentos dos outros subsistemas. Os resultados das tarefas são recebidos por esses procedimentos e exibidos pela Interface;

– o Subsistema de Modelagem, responsável por traduzir as descrições dos objetos e operações de manipulação recebidas da Interface em comandos para a geração das representações internas. Deve permitir a construção de modelos CSG e dispor dos procedimentos necessários para construir a representação B-rep correspondente. Contém procedimentos que interpretam a descrição de um novo modelo ou de uma modificação e, a seguir, acionam os procedimentos do Subsistema de Representação para criar ou alterar, respectivamente, uma representação. Responde também a questões geométricas e topológicas solicitadas pelo sistema;

– o Subsistema de Representação, responsável pela manutenção, gerenciamento e acesso ao conjunto de representações internas (estruturas de dados) admitidas pelo sistema. Deve também incluir funções para armazenamento da descrição dos objetos e composições em bases de dados permanentes. Cada representação admitida deverá possuir uma estrutura de dados e procedimentos específicos para a manipulação dessa estrutura. As informações das representações internas poderão ser lidas por procedimentos dos outros subsistemas, mas serão criadas e modificadas apenas pelos procedimentos do Subsistema de Representação;

– o Subsistema de Geração de Malha, responsável pela geração da malha de elementos finitos a partir da descrição geométrica da fronteira, fornecida pela representação B-rep. Obtém-se como resultado a divisão do domínio do problema (o modelo) em um conjunto de subdomínios (os elementos), respeitando as fronteiras e as interfaces do domínio inicial. Este processo de discretização implica o aparecimento de um certo número de nós (vértices) em que serão definidos

os graus de liberdade utilizados nas equações dos elementos finitos durante a etapa de processamento.

De uma maneira geral, os usuários interagem com o modelador para descrever e editar objetos sólidos por meio de uma interface gráfica interativa, que constitui o Subsistema de Interface. Os comandos recebidos pela Interface são traduzidos em solicitações para o Subsistema de Modelagem, que implementa a técnica CSG e é responsável pela execução das operações de manipulação da forma. Para tal, esse subsistema interage com o Subsistema de Representação, que acessa e altera, quando necessário, as estruturas de dados mantidas pelo sistema. Uma vez criado um modelo, o usuário poderá solicitar a geração da malha de elementos finitos sobre ele, tarefa realizada pelo Subsistema de Geração de Malha. A composição de sólidos gerada pode então ser armazenada sob a forma de uma base de dados neutra contendo a descrição geométrica da composição, bem como a malha de elementos finitos correspondente. Programas aplicativos e interfaces externas poderão, por intermédio dessa base, acessar tais informações com vistas a um processamento específico.

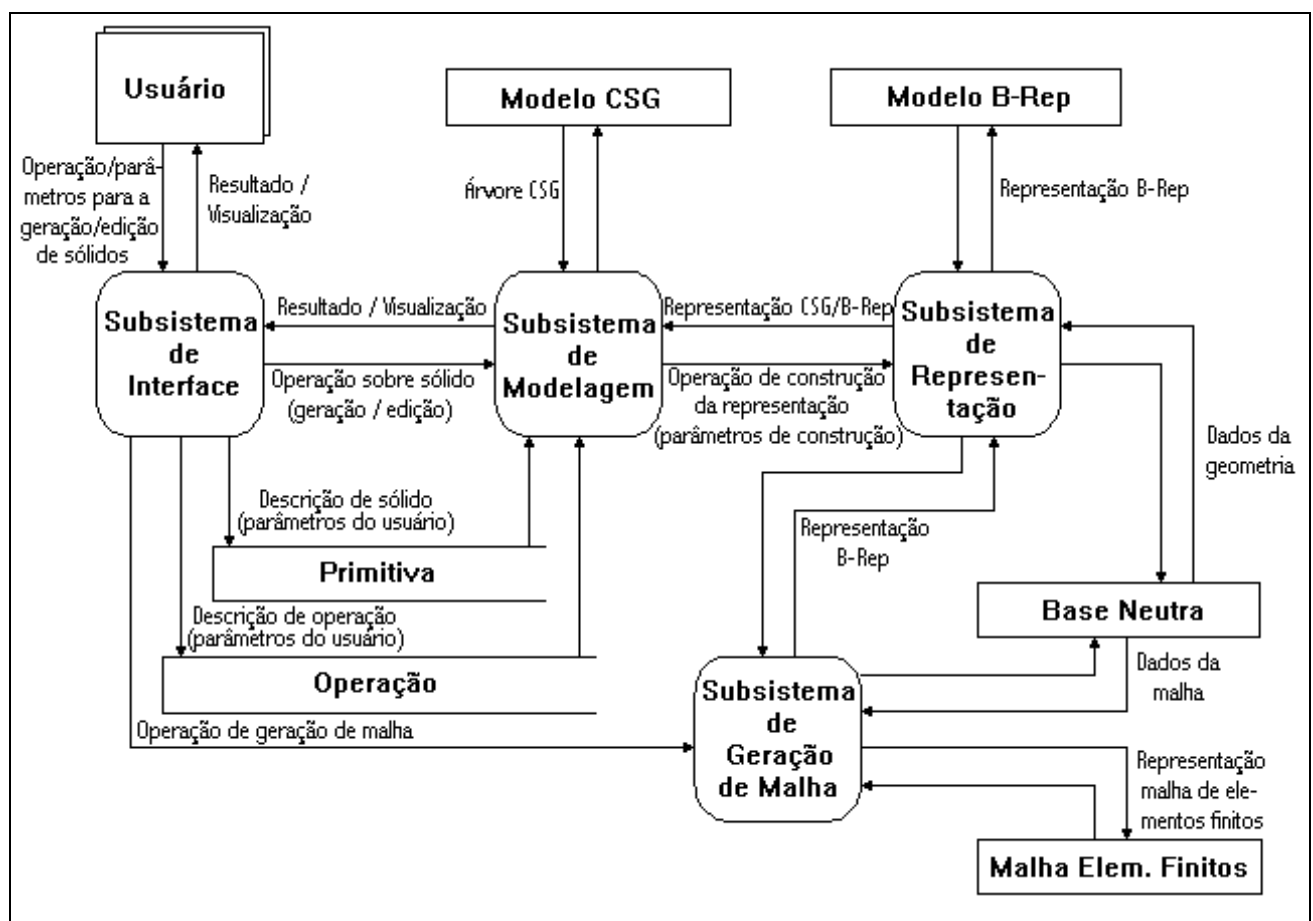



Figura II.1 – Estrutura funcional do modelador

II.3.2 – ESPECIFICAÇÃO DE REQUISITOS

Esta seção apresenta em forma textual os requisitos gerais a serem atendidos pelo modelador, basicamente a necessidade de utilização da programação orientada para objetos, de padronização de bases de dados e de parametrização do sistema. Requisitos relacionados aos subsistemas de Interface, Modelagem, Representação e Geração de Malha serão relacionados nos capítulos específicos para estes subsistemas.

II.3.2.1 – UTILIZAÇÃO DE PROGRAMAÇÃO ORIENTADA PARA OBJETOS

A filosofia de Análise e Programação Orientadas para Objetos (AOO e POO) [Boo91, Coa96] deverá ser utilizada em todo o projeto, pois suas características modulares proporcionam consideráveis vantagens no desenvolvimento de softwares de elementos finitos [For90, Sil93] quando comparadas ao enfoque tradicional (procedural) [Gan83, Pre87]. Entre os principais objetivos da POO destacam-se o de ser portátil, reutilizável, confiável e fácil de manter programas complexos, alcançados pela introdução de conceitos como a abstração e o encapsulamento de dados (implantados por meio de objetos, classes e métodos), herança (hierarquia de classes) e polimorfismo (possibilidade de usar interfaces idênticas com diferentes implementações). Detalhes sobre estes conceitos e sua utilização podem ser obtidos em referências clássicas de AOO e POO [Wie91, Boo91, Coa96, Rum97, Boo00].

Como os códigos de elementos finitos demandam uma linguagem que trabalhe eficientemente do ponto de vista numérico, e os esquemas de representação requerem estruturas de dados complexas, a linguagem C++ [Mon94, Per94, Koe96, Str97] juntamente com a biblioteca STL de estruturas genéricas de dados [Mus96] foram escolhidas para o desenvolvimento do modelador, uma vez que permitem combinar as vantagens da POO com a eficiência computacional do C [Ell93]. O ambiente de desenvolvimento de aplicativos para *Windows Borland C++ Builder* [Cal97, Mia97, Rei98] está sendo utilizado como agilizador do processo de construção de janelas do modelador e a biblioteca gráfica *OpenGL* [Wri96] foi escolhida para automatizar rotinas gráficas. Prevê-se para o futuro uma adaptação para a plataforma UNIX, utilizando provavelmente o *Kylix* [Gar00], um ambiente de desenvolvimento semelhante ao *Delphi* e ao *C++ Builder* que a *Borland* está desenvolvendo para o ambiente *Linux*.

II.3.2.2 – INTERCÂMBIO DE DADOS COM OUTROS PROGRAMAS

Além das bases de dados geradas para uso do próprio sistema, o modelador deverá possibilitar a geração de uma base de dados neutra para o intercâmbio de dados com programas processadores e pós-processadores de cálculo de campos eletromagnéticos. Mais conhecida por Arquivo Neutro, esta base padroniza os formatos de transferência de dados, melhorando a compatibilidade e a portabilidade dos dados e arquivos envolvidos. A figura II.2 ilustra a participação do arquivo neutro no intercâmbio de dados entre as diversas etapas da análise de um modelo. Da mesma forma, programas aplicativos e interfaces externas podem recuperar informações da base de dados neutra visando a um processamento específico. Maiores detalhes serão apresentados no capítulo VII.

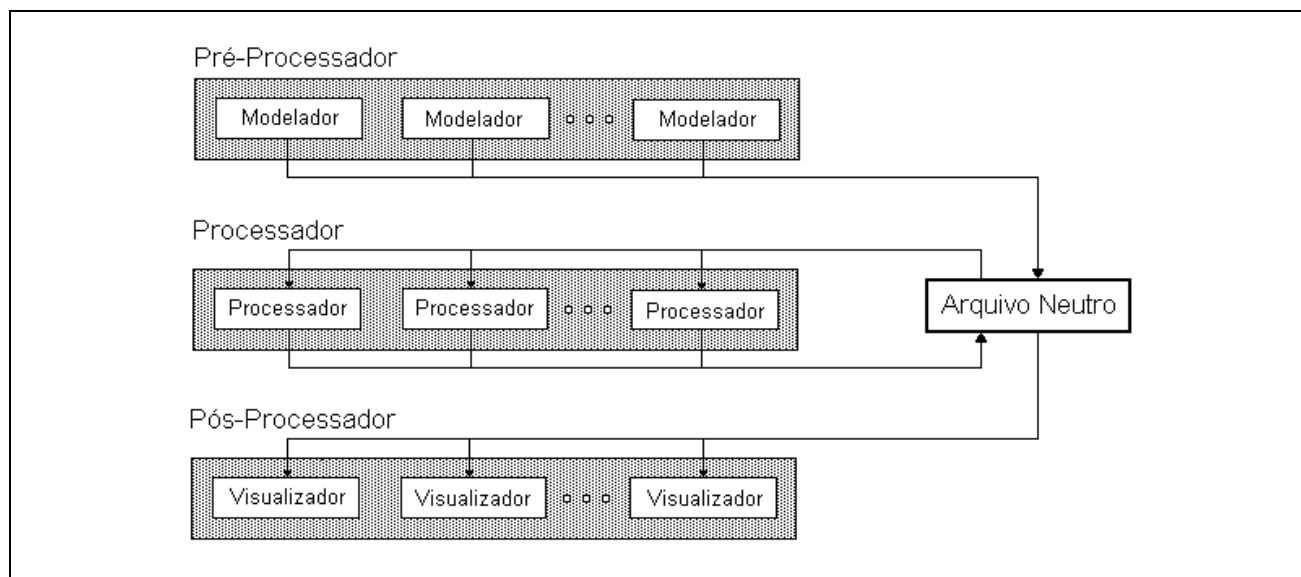



Figura II.2 – A participação do arquivo neutro no intercâmbio de dados entre as etapas da análise de um problema

II.3.2.3 – PARAMETRIZAÇÃO DO SISTEMA

A fim de obter um produto otimizado, diversos protótipos com diferentes especificações de formato são geralmente construídos e analisados, para posterior ajustamento da forma. Torna-se necessário, portanto, que modeladores voltados para otimização satisfaçam alguns requisitos, tais como: dispor de melhores ferramentas para a construção e edição do modelo, uma vez que o usuário nem sempre está certo quanto à obtenção de uma configuração final satisfatória; permitir a redefinição do tamanho, bem como da localização de todos os elementos e/ou componentes geométricos do projeto; possibilitar o reaproveitamento de projetos anteriores em um novo projeto.

Isso é possível a partir da parametrização de sistemas e geração de modelos genéricos – modelos representando uma família de objetos diferentes que compartilham entre si as mesmas expressões topológicas, apesar de possuírem geometrias distintas [Sha95]. Sistemas parametrizados armazenam a geometria do objeto por meio de parâmetros, possibilitando um dimensionamento variável: a partir de um conjunto específico de parâmetros, componentes específicos de uma família podem ser obtidos.

O projeto parametrizado está-se tornando uma metodologia útil, não só por aumentar a flexibilidade no processo de projeto, definindo expressões geométricas ao invés de especificar dimensões concretas do objeto, mas também por possibilitar a geração de bibliotecas de componentes com formato padrão. O uso de expressões geométricas permite também descrever dependências entre componentes de um objeto ou entre objetos.

A abordagem de projeto parametrizado a ser posteriormente incorporada ao modelador é a construtiva [Sol94]. Além de ser uma forma natural de trabalho, na qual o projetista utiliza operações e expressões de modelagem à medida que vai construindo o objeto, ela segue uma estratégia semelhante à abordagem CSG, encorajando a decomposição do problema em subproblemas. Constituindo um recurso de apoio ao desenvolvimento de projeto, a abordagem construtiva a ser implementada será simples e não suportará expressões circulares (que precisam ser solucionadas simultaneamente).

II.3.3 – A Análise Orientada para Objetos

A análise orientada para objetos do modelador foi desenvolvida seguindo a abordagem sugerida por Coad e Yourdon [Coa96]. A partir da especificação de requisitos gerais e específicos para cada subsistema, foram identificadas as principais classes do sistema e as relações de dependência entre elas. A seguir, foram detalhados os atributos e os principais serviços relacionados a cada classe. Com base nas estruturas obtidas a partir das relações de dependência e nos requisitos especificados foi possível definir os assuntos tratados pelo sistema e agrupá-los em relação aos módulos funcionais definidos. O resultado da estruturação geral de assuntos é apresentado na figura II.3. O detalhamento da análise será apresentado nos capítulos específicos para cada subsistema, que apresentará diagramas de classes e relações de dependência utilizando recursos de engenharia reversa disponíveis no *Rational Rose* [Qua98].

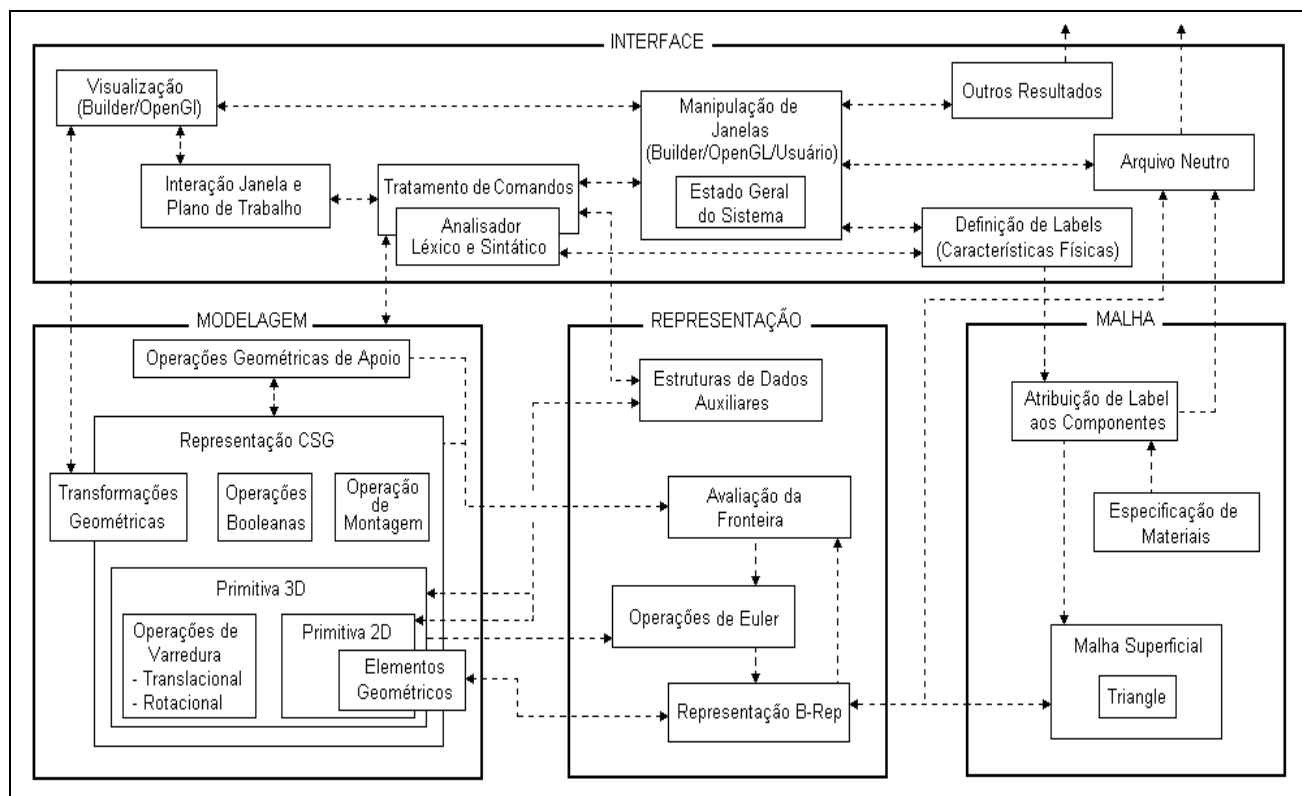



Figura II.3 – Estruturação de assuntos do modelador em desenvolvimento

II.3.4 – SOLUÇÕES ORIENTADAS PARA OBJETOS

O projeto de softwares orientados para objetos não é uma tarefa trivial. O projeto de software reutilizável orientado para objetos é ainda mais difícil, pois é necessário atender não só ao problema específico atual, mas também ser genérico o suficiente para incorporar problemas e requisitos futuros. Todo construtor de software deve procurar evitar a necessidade de “reinventar a roda”. Caminhando nessa direção, especialistas em projeto orientado para objetos sugerem diversas soluções genéricas, adequadas e eficientes para determinados padrões de comportamento, denominados *patterns* [Gam95]. Tais padrões ajudam o projetista de software a resolver problemas específicos de projeto baseados em experiências bem sucedidas no passado, obtendo soluções mais flexíveis, elegantes e reutilizáveis. Facilitam também a documentação e manutenção de sistemas.

Alguns padrões de comportamento descritos por Gamma et al. [Gam95] mostraram-se interessantes para o projeto deste modelador e foram a ele adaptados. Os padrões de uso geral são descritos a seguir, e os específicos serão apresentados em seus respectivos capítulos.

As estruturas de dados mais utilizadas em um modelador de sólidos que possua as representações CSG e B-rep são listas e árvores. É interessante ter uma forma de manipular esses objetos sem expor sua estrutura interna. Além disso, é desejável "caminhar" nessas estruturas em diferentes seqüências, dependendo do que deve ser feito. Para isso, é possível retirar da estrutura a responsabilidade pelo acesso e "caminhamento", criando agentes de iteração denominados *Iterators*, como mostrado na figura II.4, em notação UML [Boo00]. Esse agente e a estrutura de dados estão acoplados: o *Iterator* define uma interface para ter acesso aos elementos, acompanha o elemento corrente em cada estrutura e sabe exatamente quais elementos já foram percorridos. A separação entre o mecanismo de travessia e a estrutura permite que diferentes agentes de iteração sejam definidos para "caminhamentos" diferentes, sem que seja necessário enumerá-los na interface da estrutura. Uma classe abstrata, *AbstractStructure*, que fornece uma interface padrão para a manipulação das estruturas é definida, bem como uma classe abstrata para o *Iterator*, definindo uma interface para iteração padrão. Subclasses *Iterator* concretas para diferentes implementações da estrutura podem ser, então, definidas. Como resultado, o mecanismo de iteração torna-se independente da classe concreta da estrutura e suporta polimorfismo.

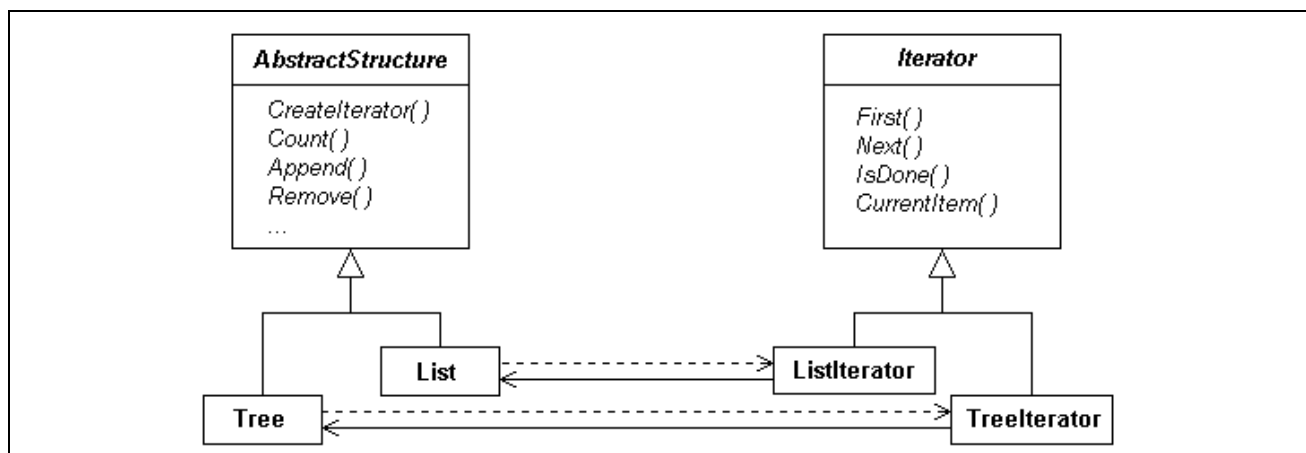



Figura II.4 – O padrão *Iterator* fornecendo uma interface uniforme para manipular diferentes estruturas de dados

Diferentes tipos de dados são manipulados como componentes em listas e árvores pelo Subsistema de Representação. Para evitar duplicação de código nesses componentes, comportamentos comuns podem ser fatorados e colocados em classes comuns, fornecendo recipientes de armazenamento (*containers*) adaptáveis, que implementam partes invariantes de um algoritmo uma única vez e repassam para as subclasses a implementação de comportamentos que podem variar. Isso é o que o padrão *Template* deve fazer: definir um esqueleto de um algoritmo em uma operação, repassando alguns passos para as subclasses, que os redefinem sem alterar a estrutura do algoritmo.

Tipos definidos como instâncias de classes *Template* podem ser usados em qualquer lugar onde tipos ordinários possam ser usados. Da mesma forma, métodos *Template* podem ser usados para definir rotinas genéricas. Como mostrado na figura II.5, um método *Template* define um algoritmo em termos de operações abstratas que podem ser sobrepostas para fornecer comportamentos concretos. Definindo alguns passos de um algoritmo usando operações abstratas, o método *Template* fixa a ordem de execução, mas permite que as subclasses variem tais passos para se adequar a suas necessidades. De todo modo, os projetistas devem estar atentos para reduzir o número de operações que devem ser sobrepostas.

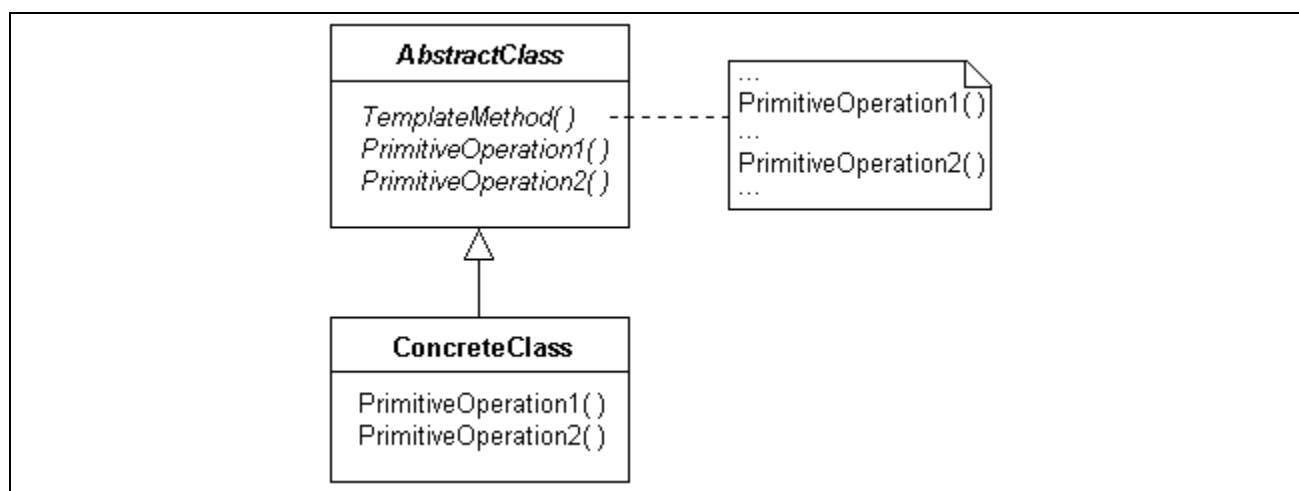



Figura II.5 – Redefinição de operações no padrão de projeto para *Template* [Gam95]

A idéia por trás de *iterators* e *templates* aqui apresentada foi incorporada ao ANSI C++, já existindo tipos de dados definidos diretamente utilizando esta filosofia [Mus96], porém, para situações nas quais este padrão não pode ser utilizado diretamente, será necessário desenvolver mecanismos como os apresentados acima.

II.4 – Características Gerais de Implementação

O modelador está sendo implementado em microcomputadores *Pentium* gerenciados pelo *Windows 95* ou *NT*. Como dito anteriormente, utiliza os conceitos de orientação para objetos, o ambiente de desenvolvimento *Borland C++ Builder*, a biblioteca de estruturas genéricas de dados STL e a biblioteca gráfica *OpenGL*. Seus subsistemas estão sendo implementados simultaneamente, com a participação de quatro alunos bolsistas de iniciação científica, cada um trabalhando em um subsistema. Detalhes sobre a implementação dos subsistemas serão apresentados em seus capítulos.

À medida que programas vão crescendo em tamanho e complexidade, torna-se necessário particioná-los em módulos. Uma preocupação constante durante a implementação do modelador tem sido com a sua modularização, uma vez que ela aumenta a legibilidade e a portabilidade do sistema [Pre87], além de facilitar sua posterior manutenção. Para softwares grandes e complexos, como é o caso do modelador, o ideal é utilizar a estratégia "dividir para conquistar" desde a fase de projeto, partindo para a definição de módulos e a utilização da programação modular.

Além de vencer barreiras de complexidade pela resolução e integração de diversos problemas menores, a programação modular traz várias outras vantagens [Sta98]: permite distribuir trabalho, tornando possível que diversas pessoas atuem simultaneamente no desenvolvimento; permite reutilizar módulos, diminuindo o volume de trabalho a ser realizado; reduz o tempo de compilação, uma vez que somente os módulos que sofreram alteração no código-fonte precisam ser recompilados; permite o desenvolvimento incremental de programas, no qual versões mais completas e robustas vão sendo criadas, substituindo as anteriores e prestando algum serviço útil adicional; permite investir na criação de um acervo de componentes utilizados com freqüência nos programas desenvolvidos pela área; permite deixar o aprimoramento do desempenho para uma época mais oportuna. Por outro lado, o projeto modular traz problemas: para obter um programa composto por diversos módulos é necessário que as interfaces entre estes módulos sejam bem definidas e respeitadas pelos desenvolvedores.

O elemento essencial da programação modular é a interface, mecanismo pelo qual é realizada a troca de dados, comandos e eventos entre dois módulos de um software. Para que não ocorram erros de comunicação é necessário que a sintaxe e a semântica transmitidas sejam coerentes com a esperada e que o estado do servidor esteja consistente com os dados transmitidos e com a intenção de uso por parte do cliente. As interfaces devem conter somente declarações e implementações na forma de macro e de código *inline*. As declarações contidas nas interfaces entre módulos são basicamente declaração de classes, de métodos e de tipos, bem como dados globais e funções não vinculadas a classes.

II.4.1. – A Importância da Independência Funcional entre os Módulos

O principal objetivo da programação modular é buscar a independência funcional entre os módulos. A geração de módulos independentes facilita as atividades de manutenção e teste, pois permite diminuir os efeitos secundários causados por alterações no projeto e código, reduzindo a propagação de erro e possibilitando o desenvolvimento de módulos reutilizáveis.

A independência funcional está diretamente relacionada com os conceitos de encapsulamento, coesão e acoplamento. O encapsulamento é a propriedade de esconder a implementação de um módulo, protegendo-o contra uso ou alteração acidental e viabilizando o seu acesso exclusivamente por intermédio da interface exportada. A coesão é uma extensão natural do conceito de encapsulamento e mede a interdependência semântica entre os elementos do módulo, ou seja, sua força funcional: um módulo coeso deve realizar idealmente uma única tarefa e requerer pouca interação com procedimentos realizados por outras partes de um programa. Durante o projeto de módulos, deve-se sempre buscar uma alta coesão.

Já o acoplamento é uma medida da interdependência relativa entre módulos. O acoplamento depende do volume de elementos que compõem a interface e da forma como é estabelecida a interface entre esses elementos. Deve-se procurar o menor acoplamento possível entre os módulos, o que pode ser obtido: reduzir ao mínimo o número de conectores (protótipos, variáveis globais, classes, etc.) contidos em uma interface; reduzir ao mínimo o número de elementos contidos em cada conector; diminuindo a complexidade dos dados; estabelecendo uma comunicação mais direta e correta, evitando regras, restrições, ordenações, valores especiais, etc.; evitando efeitos secundários, que tendem a dificultar a manutenção, tais como alteração de variáveis globais, alteração do conteúdo de arquivos e de tabelas. Uma conectividade simples entre os módulos resulta em um software mais fácil de entender e menos sujeito a efeitos colaterais, decorrentes de propagação de erros.

II.4.2 – Regras Adotadas para a Obtenção de Programas Modularizados

Não existe um consenso quanto à definição de módulo. Para este projeto, um módulo é uma unidade lógica independentemente compilável, formada basicamente pelo código-fonte, seu arquivo-cabeçalho e mais algum arquivo de tabelas ou dados relacionado. Do ponto de vista lógico, um módulo implementa, sempre que possível, um único conceito – uma classe, um assunto ou um componente. Do ponto de vista físico, um módulo é uma unidade de compilação em separado.

Visando obter programas bem modularizados, Staa estabeleceu algumas regras para programar módulos em C/C++ [Sta98] que nortearam a elaboração de um esqueleto para os módulos deste projeto. Cada módulo é composto por um módulo de definição (.h) e um módulo de implementação (.cpp). Para o módulo composto pela função principal foi associado um módulo de definição contendo tipos de dados, contantes e variáveis globais. A estrutura básica dos módulos de definição e de implementação bem como as regras gerais de implementação assumidas durante o desenvolvimento são detalhadas no Apêndice I – Padronização Adotada Durante o Desenvolvimento do Modelador.

II.5 – Características Gerais da Documentação Produzida

A sistemática de documentação utilizada também foi baseada no conjunto de regras de especificação e documentação estabelecidas por Staa [Sta98], porém bastante simplificada. Ao invés de utilizar marcadores para posterior referência e duplicar informações em módulos de definição e implementação, como propõe Staa, optou-se por colocar somente as informações consideradas essenciais no código e incluir links para a documentação do sistema, montada em formato HTML. Isto diminui a redundância de informações e o risco de inconsistências geradas pela atualização parcial das informações duplicadas em módulos de definição e implementação. A padronização definida para a documentação do sistema e do código é apresentada no Apêndice I.

A documentação padronizada para o código é formada basicamente por cabeçalhos predefinidos para arquivos, classes, métodos ou funções, e comentários para atributos, variáveis e definições de tipos de dados. Além de informações técnicas, todos os arquivos do código devem também conter um conjunto de informações gerenciais –identificação do arquivo, descrição de seu conteúdo e histórico de alterações – que facilitam administrar a evolução do código.

Além da padronização adotada para o código, a documentação *on-line* envolve todas as informações relacionadas ao desenvolvimento do modelador, incluindo o projeto dos sub-sistemas, a distribuição de assuntos dentro de cada subsistema, o detalhamento das bases de dados manipuladas, etc. Criou-se desta forma uma árvore de arquivos HTML documentando cada componente do modelador, acessível a partir do endereço *http://www.cpdee.ufmg.br/~gopac/gsm.*

"*Complicar o que é simples, é comum.*
Tornar simples o que é complicado, é criatividade."
Jennifer Dallegrave

III – O SUBSISTEMA DE INTERFACE

A utilização de um sistema interativo é, em sua essência, um processamento controlado por escolhas do usuário em momentos convenientes. Fundamental em um sistema interativo, a interface é o elemento responsável por estabelecer diálogo com o usuário. A forma desse diálogo e a linguagem de interação possuem importância vital: uma interface mal planejada pode comprometer a eficiência do usuário e levá-lo a rejeitar o sistema como um todo O usuário de um sistema gráfico interativo espera poder contar com uma ferramenta eficiente, robusta, que facilite seu trabalho e lhe dê liberdade para manipular os objetos envolvidos. A interface merece, portanto, uma atenção especial no desenvolvimento de sistemas.

Este capítulo descreve o Subsistema de Interface, responsável por criar e gerenciar todos os recursos gráficos, bem como interagir com o usuário, descrevendo operações que ativam procedimentos dos outros subsistemas. A seção III.1 relaciona as características de projeto mais importantes, envolvendo especificação de requisitos, modelamento e soluções orientadas para objetos. A seção III.2 descreve a implementação do principais assuntos relativos à interface, incluindo o ambiente criado para a construção de modelos, o tratamento de comandos e os recursos disponíveis para visualização. A seção III.3 apresenta os principais componentes disponíveis na interface do modelador.

III.1 – PRINCIPAIS CARACTERÍSTICAS DE PROJETO

O projeto da interface foi desenvolvido levando em consideração alguns cuidados descritos na literatura e tendo em mente a necessidade de retratar o sistema do ponto de vista do usuário. A especificação de requisitos procurou apresentar o sistema na forma como ele estará disponível para uso, descrevendo suas características e funções, definindo *o que* o modelador fará e *como* o usuário o utilizará. A partir da especificação foi definida a estrutura interna da interface. A seguir, as principais classes envolvidas foram identificadas e agrupadas, definindo assim os assuntos a serem tratados. Entre estes, alguns mereceram um estudo mais detalhado, em busca de soluções orientadas para objetos a serem adotadas pela implementação.

III.1.1 – REQUISITOS RELACIONADOS AO PROJETO DE INTERFACES

Existem muitos fatores a serem considerados no projeto de interfaces com o usuário. Além das operações disponíveis, fatores como a concepção do sistema pelo usuário, a organização de menus e barras de tarefas, a resposta a entradas e erros, a organização e apresentação de resultados, bem como a documentação de apoio, deverão ser analisados cuidadosamente.

O projeto de uma interface inicia-se com a elaboração do modelo conceitual a ser apresentado para o usuário. Este modelo descreve *o que* o sistema deve fazer e que operações gráficas estão disponíveis, especificando objetos e operações a serem realizadas sobre os objetos. As informações devem ser apresentadas na linguagem da aplicação e somente conceitos já familiares devem ser empregados. O modelo conceitual deve ser tão simples e consistente quanto possível: modelos complicados, contendo objetos e operações definidas de forma diferente em contextos diferentes, dificultam seu entendimento e utilização. Deve-se ainda procurar reduzir o número de objetos e operações gráficas no modelo – o que facilita sua aprendizagem – mas sem simplificá-lo ao extremo, o que dificultaria sua aplicação.

Para permitir a comunicação entre o usuário e o ambiente é necessária uma linguagem de interação, usualmente denominada linguagem de comandos, a qual deve possuir algumas características especiais, como: ser bastante simples e empregar termos relacionados à área de aplicação, para que o usuário possa aprendê-la de forma natural; possuir uma sintaxe concisa e coerente, de forma que as ações sejam tratadas sempre de um mesmo modo e aplicadas a todos os objetos; ser estruturada de forma que o usuário não necessite desviar a atenção entre dispositivos ou áreas da tela gráfica; ser responsável por direcionar as ações, acusando o recebimento de entradas e avisando qual o tipo de ação esperada. Além dessas, outras características devem ser observadas: os parâmetros – informações adicionais requeridas pela ação – deverão ser verificados quanto à seqüência e ao conteúdo, de maneira que erros de sintaxe possam ser detectados e corrigidos sem que fracasse a ação; a seqüência de parâmetros em um comando deve ser a mais uniforme possível; no tratamento de erros, bons diagnósticos e mensagens claras devem explicar o que ocorreu e o que deve ser feito para corrigir a situação; facilidades de ajuda de operação deverão ser incluídas, em diferentes níveis; sempre que possível, haverá disponibilidade para cancelamento de ações em qualquer ponto de uma seqüência de parâmetros.

As opções de processamento devem ser apresentadas para o usuário em menus, o que permite não só reduzir a quantidade de memorização necessária, mas também evita a seleção daquelas não disponíveis em uma dada situação. As opções do menu são geralmente selecionadas

pelo *mouse* ou por teclas de atalho definidas. Na prática, menus concisos e com menos opções são mais efetivos, uma vez que reduzem o tempo de procura por uma opção e ocupam menos espaço na tela. Menus longos devem ser divididos em submenus e reorganizados, formando uma estrutura multinível. Na maioria dos sistemas eles são colocados sempre na mesma posição, habituando o usuário a fazer a seleção em uma posição fixa. Outra possibilidade é utilizar menus móveis *pop-up*, que são abertos na posição corrente do cursor, minimizando movimento dos olhos e da mão. Um item de menu pode ser apresentado por um nome e/ou por um ícone. O ícone possui a vantagem de ocupar menos espaço e ser mais rapidamente reconhecido do que uma descrição textual, motivo pelo qual é normalmente utilizado em barras de ferramentas, forma alternativa de apresentar as opções disponíveis.

Outro fator importante a ser considerado no projeto de interfaces é como o sistema responde ou realimenta o usuário. Para toda ação do usuário deverá haver uma resposta por parte do sistema – uma realimentação –, de preferência condensada em um único local. A realimentação auxilia o usuário na operação do sistema, acusando o recebimento de dados e comandos, enviando mensagens ao usuário, sinalizando a identificação de seleções e avisando o que está fazendo a cada passo. Ela é particularmente importante quando o tempo de resposta é grande: nesta situação, mensagens devem informar o progresso do processamento, evitando a falsa impressão de falha do sistema. As mensagens devem ser claras o suficiente para que haja pouca chance de serem negligenciadas, mas, por outro lado, não devem ser tão destacadas a ponto de distrair a atenção do usuário. As formas de realimentação mais comumente utilizadas são o cursor piscante, indicando que o sistema está ocioso, e o eco de teclado, que apresenta imediatamente na tela o que foi digitado, permitindo verificar e corrigir a digitação. A realimentação também pode ser feita por áudio, com a vantagem de não ocupar espaço e não necessitar da atenção do usuário voltada para a tela.

A informação apresentada ao usuário inclui uma combinação de imagens, menus, mensagens e outras formas de diálogo geradas pelo sistema. Existem muitas formas de dispor estas informações de saída para o usuário, ficando a cargo do projetista verificar qual é o melhor formato para obter o efeito visual desejado. O *layout* da tela inclui pelo menos três elementos básicos, que ocupam porções fixas ou redimensionáveis: a área de trabalho, a área de menus e a área para apresentar realimentações e gerar entrada de dados. Uma boa flexibilidade na organização da tela pode ser obtida permitindo a sobreposição de janelas. Neste caso, um conjunto de operações para criar, apagar e reposicionar áreas de janelas deve ser fornecido. O *layout* de tela deve evitar uma aparência desorganizada, mantendo áreas simples e de fácil compreensão. O uso de cores e estilos diferentes de linha permitem melhorar o efeito visual.

III.1.2 – REQUISITOS RELACIONADOS À INTERFACE DO MODELADOR

Utilizando uma interface gráfica interativa, que constitui o Subsistema de Interface, o usuário poderá gerar e editar objetos sólidos da maneira que lhe for mais conveniente. Para tal fim, o Subsistema de Interface deve: facilitar e agilizar o trabalho do usuário; fornecer formas variadas para definir sólidos e dispor de operações para manipulá-los; descrever tarefas e ativar outros subsistemas para executá-las; evitar a execução de operações incorretas; manter o usuário informado sobre o estado geral do sistema; receber e apresentar os resultados obtidos de tarefas executadas; prover recursos para uma melhor visualização do modelo; possibilitar a comunicação do modelador com outras aplicações.

A tela principal do modelador deverá ser composta por um menu, uma área gráfica de trabalho, uma área de comunicação da interface com o usuário, uma área para entrada de comandos e parâmetros e uma área apresentando o estado corrente do sistema. Barras de ferramentas contendo ícones de comandos deverão estar disponíveis, com apresentação e posicionamento a critério do usuário. Todas as operações do modelador poderão ser ativadas pela seleção de opções do menu ou ícones disponíveis nas barras de ferramentas e, quando conveniente, por linhas de comando textuais. Neste último caso, a interface deverá dispor de um interpretador – um analisador léxico e sintático – para verificar a validade do comando ou expressão digitada. A Interface deverá ainda impedir, sempre que possível, a execução de operações incorretas.

Os principais recursos que deverão estar disponíveis relacionam-se a posicionamento, identificação, definição e manipulação dos elementos geométricos. O posicionamento de um elemento no modelo pode ocorrer de diversas formas: pela localização do cursor (*mouse*) na área gráfica; pela entrada, via teclado, de coordenadas absolutas, polares ou relativas (Δx, Δy e Δz em relação a um ponto anteriormente fornecido); pelo fornecimento de ângulos e distâncias, diretamente via teclado ou pela realização automática de cálculos a partir de pontos fornecidos pelo *mouse*. A identificação e seleção de um elemento do modelo pode ser feita pelo posicionamento do cursor sobre ele ou pela especificação de seu nome, se este existir. Em ambos os casos, será necessária a existência de uma estrutura de dados aos quais se associe o elemento. A identificação pelo cursor implicará a necessidade de comparação das coordenadas do cursor com as coordenadas dos elementos existentes no modelo, o que aumentará um pouco o tempo de resposta do sistema. Nesta situação e em outras em que o sistema necessite de um maior tempo de processamento, o usuário deverá ser alertado quanto à necessidade de espera e à evolução do processamento. Um recurso de *back-up* também deverá estar disponível a qualquer tempo, possibilitando ao usuário explorar as capacidades do sistema com segurança.

A construção de um modelo tem início com a geração de uma primitiva. A princípio, estarão disponíveis nove tipos de primitivas sólidas: blocos, prismas, pirâmides, cilindros, cones, esferas, hemisférios, elipsóides e toros. O usuário poderá gerar outras primitivas definindo perfis 2D e aplicando operações de varredura translacional e rotacional. Estes perfis poderão ser definidos por uma seqüência de primitivas 2D coplanares e sem auto-interseção, formando polilinhas compostas por linhas e arcos. Um perfil poderá envolver outros, desde que não ocorra interseção entre eles. Para a varredura rotacional, poderão ser definidos perfis abertos ou fechados. O eixo de rotação deverá ser externo e coplanar ao perfil. Visando garantir a geração de sólidos, as varreduras translacional simples e cônica serão sempre realizadas sobre um perfil fechado e em qualquer direção não coplanar ao perfil.

Para manipulação, os principais recursos disponíveis serão: copiar, transladar, rotacionar, refletir e escalar. Estes recursos poderão ser aplicados sobre as primitivas definidas, bem como sobre componentes do modelo, permitindo alterar posição, orientação e tamanho. Para obter sólidos mais complexos, as primitivas sólidas geradas poderão ser combinadas por meio das operações booleanas regularizadas de união, interseção e diferença. Este processo de construção seguirá a filosofia CSG, gerando uma árvore de definição do objeto composta por primitivas sólidas e operações. Visando agilizar o processo de construção e evitar redundâncias na definição de sólidos, poderão ser utilizadas, alternativamente, composições em substituição às primitivas. Denominadas agrupamentos, estas composições nada mais serão do que sub-árvores CSGs formadas por primitivas e operações. Também estará disponível a operação de montagem, que permitirá definir interfaces – fronteiras internas – entre componentes do modelo. Visando facilitar a manipulação e o posicionamento de primitivas, será possível escolher o plano de trabalho mais adequado, passando de um sistema de coordenadas global para um sistema de coordenadas local, definido pelo usuário.

O Subsistema de Interface deverá transmitir ao usuário a sensação de estar em contato direto com o modelo em construção, e para isso será fundamental a existência de uma forma efetiva de interagir com o modelo, possibilitando "caminhar" ao seu redor, bem como analisá-lo sob todos os ângulos. Para atingir esse objetivo, além da imagem fio-de-arame do modelo deverão estar disponíveis pelo menos os seguintes recursos de visualização: a eliminação de linhas e superfícies escondidas, que possibilitará "limpar" e "descongestionar" o desenho; o *zoom*, que permitirá ampliar ou reduzir o modelo apresentado na área de trabalho; o corte, que mostrará o modelo por dentro; a projeção em qualquer ângulo solicitado pelo usuário; o deslocamento sucessivo da câmera de visualização, tanto em torno como para dentro e para fora do modelo, o que, combinado com os recursos anteriores, transmitirá ao usuário a sensação de ter o modelo nas mãos. Outros recursos, como a iluminação do modelo, a inclusão de textos e de cotas, serão futuramente desenvolvidos.

III.1.3 – O MODELAMENTO OBTIDO

A especificação de requisitos da interface permitiu identificar as principais classes e relações de dependência entre elas, ilustradas na figura III.1. Uma análise um pouco mais detalhada levou à obtenção dos atributos e métodos essenciais de cada classe. A partir do resultado obtido, esquematizado no quadro III.1, foi possível identificar os assuntos tratados pelo Subsistema de Interface bem como relacioná-los entre si e com o demais assuntos do sistema. O resultado é apresentado na figura III.2.

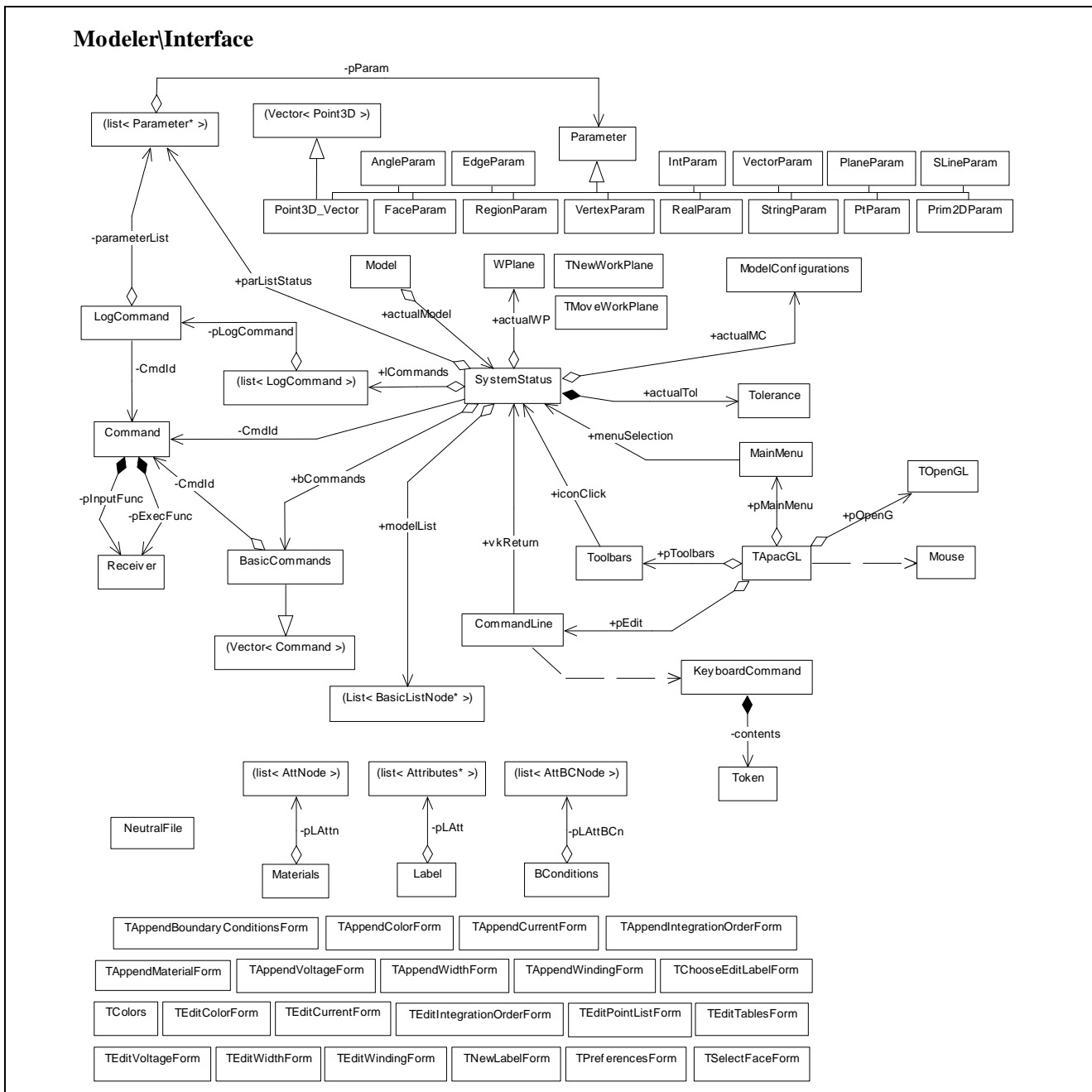



Figura III.1 – Diagrama das principais classes e relações de dependência entre elas

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
TApacGL: formulário básico, principal do modelador	
Menu, itens de menu, botões, barras de ferramentas, barra de status, linha de comando, área de comunicação com o usuário (memo), área gráfica e caixas de diálogo	Criação, seleção, ativação e redimensionamento de componentes, controle do *mouse* e do teclado, conversão de coordenada, disparo de comando
TOpenGL: área gráfica, onde será apresentado o modelo	
Largura, altura, posição, cor de fundo, coordenadas opostas extremas, tipo de cursor	Repintura, redimensionamento, movimentação e posicionamento do *mouse*
Mouse: controle de entrada de dados pelo *mouse*	
Botão pressionado, posição(x,y)	Identificação e interpretação de click em botão, captura de coordenada, conversão entre coordenadas
KeyboardCommand: controle de entrada de dados pelo teclado	
Tipo de entrada, posição inicial, posição atual, conteúdo	Pressionamento de teclas, identificação de caracter, interpretação de tokens, análise sintática da entrada
SystemStatus: controle do estado atual do sistema	
Comando atual, função para comando atual, estado atual, tipo de espera, próximo estado, pontos anteriores, tolerância atual, lista de parâmetros, modelo atual, caminho padrão para busca de arquivos	Reiniciar estados do comando, atualizar estados do comando atual, inserir parâmetro na lista, inserir ponto na lista de pontos anteriores, alterar tolerância atual, alterar modelo atual, alterar caminho
Tolerance: tolerância admitida pelo sistema	
Valor	Calcular aproximação, verificar se foi satisfeita
Command: comando disponível no modelador	
Nome, identificador, funções relacionadas à execução e à entrada de parâmetros, estado inicial, estado final	Direcionar execução, fornecer atributos
BasicCommands: tabela de comandos disponíveis	
Vetor de comandos disponíveis, do tipo Command	Gerenciar os comandos disponíveis
LogCommand: comando executado ou em execução	
Identificador, lista de parâmetros	Armazenar dados sobre um comando executado
Parameter: parâmetros do sistema – possui subclasses para todos os tipos de parâmetros utilizados	
Um objeto com mesmo tipo do parâmetro a que se refere	Conversão de / para seu tipo específico
Receiver: receptor de comandos, auxiliar para tratamento homogêneo de comandos	
Nenhum	Designar função a ser executada pelo comando
Wplane: plano de trabalho	
Pontos de referência, vetor normal ao plano	Definir plano de trabalho a ser utilizado
TColors: formulário para definir cores disponíveis no sistema	
Componentes RGB da cor	Definir cores a serem utilizadas
TAppend*Form, TEdit*Form e demais formulários para definir características físicas e de material para *labels*	
Campos para digitação e seleção de características físicas	Realizar a entrada de dados para atributos relacionados às características físicas de *labels*
Label: definição dos *labels* contendo características físicas	
Nome, referência à lista de atributos	Definir, obter e gravar atributos
Attributes: atributo pertencente à lista de atributos	
Tipo, identificador	Definir, obter e gravar tipo e identificador
Materials: especificação de um material utilizado	
Nome, referência a uma lista de atributos	Definir, obter e comparar atributos
AttNode: definição de um atributo qualquer (genérico) – corresponde a uma linha da tabela de atributos	
Tipo, vetor de valores, referência a uma lista de pontos	Homogeneizar tratamento de atributos
BConditions: especificação de uma condição de contorno	
Nome, lista de atributos da condição de contorno	Definir, obter e comparar atributos
AttBCNode:	
Tipo, valores e label associados	Definir, obter e comparar atributos
NeutralFile: base de dados neutra a ser gerada para o modelo atual	
Nome, especificação do modelo, especificação do arquivo, matrizes auxiliares para montagem dos dados	Gerar arquivos, gravar cabeçalhos e dados referentes a malha, geometria e material

Quadro III.1 – Principais características das classes relacionadas

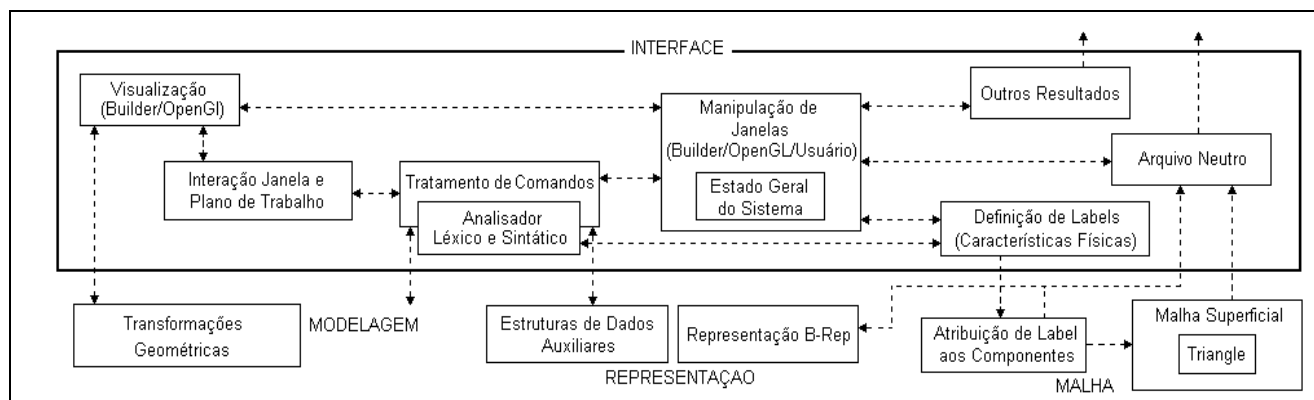



Figura III.2 – Identificação de assuntos relacionados ao Subsistema de Interface

O Controle Geral é responsável pela definição e manipulação de janelas, botões, menus e demais componentes visando ao interfaceamento com o usuário. Também gerencia a execução de comandos e controla o estado geral do sistema. Inclui diretamente o formulário básico, a área gráfica, o estado geral do sistema e a tolerância e, indiretamente, o *mouse* e o teclado.

O Tratamento de Comandos gerencia o fornecimento de comandos, assegurando a entrada de parâmetros corretos e na seqüência desejada. Engloba as atividades de análise léxica e sintática, controlando o fornecimento de dados pelo *mouse* e teclado, com os quais está diretamente relacionado. Envolve ainda os comandos disponíveis e sua lista, os comandos executados, os parâmetros fornecidos em um comando e o receptor de comandos.

A interação entre janela gráfica e plano de trabalho é o assunto da *WorkArea*, que permite ao usuário posicionar o plano de trabalho sobre um plano de interesse qualquer, dentro do modelo, e trabalhar com a tela gráfica como se estivesse sobre esse plano. Compreende basicamente a classe plano de trabalho, sobre a qual é realizada a conversão de coordenadas.

A Visualização é responsável pelos recursos disponíveis para apresentação do modelo na tela gráfica. Refere-se diretamente à área gráfica, onde o modelo é pintado, bem como à definição do modelo sob o ponto de vista da biblioteca *OpenGL*.

O assunto *Labels* controla apresentação de janelas para edição de características físicas relacionadas ao modelo, a serem atribuídas aos componentes geométricos e da malha de elementos finitos. Envolve todos os formulários para definição das características físicas – *TAppend*Form*, *TEdit*Form ...*– além das classes *Label*, *Attributes*, *Materials*, *AttNode*, , *Bconditions* e *AttBCNode*.

O assunto Arquivo Neutro gerencia o intercâmbio de dados com outros aplicativos por intermédio da base de dados neutra, que corresponde ao objeto *NeutralFile*. Os demais arquivos de dados de saída, como os que contêm as informações necessárias para recuperar um modelo previamente definido, estão incluídos no assunto denominado Outros Resultados.

III.1.4 – SOLUÇÕES ORIENTADAS PARA OBJETOS

Para assegurar a modularização e a portabilidade do modelador, é interessante garantir a independência entre os objetos da interface, como menus e botões, e os objetos do próprio modelador. Isso é possível tornando a solicitação de execução de um comando um objeto a ser tratado pelo modelador, que pode ser armazenado e usado como qualquer outro objeto do sistema. A estrutura de dados *Command* foi assim projetada, baseada no padrão *Command pattern* proposto por Gamma et ali [Gam95]. Uma visão geral dessa estrutura é apresentada na figura III.3 em notação UML [Boo00]. Uma classe abstrata *Command* declara uma interface para executar operações. Em sua forma mais simples, essa interface inclui uma operação abstrata *Execute().* Subclasses concretas de *Command* definem uma ligação entre o objeto receptor *Receiver* e uma ação, armazenando o receptor como uma variável instanciada e implementando operações *Execute()* para encaminhar o pedido; o receptor possui o conhecimento requerido para executar o pedido. Subclasses concretas de *Command* podem também ser capazes de desfazer e refazer operações ("undo" e "redo") se forem fornecidos mecanismos para reverter suas execuções. Para comandos que não podem ser desfeitos e não requerem argumentos, *templates* podem ser usados para evitar a criação de subclasses de *Command* para todo tipo de receptor e ação. Apesar da implementação realizada diferir um pouco dessa estrutura, por razões práticas, a idéia de encapsular a execução do comando foi utilizada com sucesso.

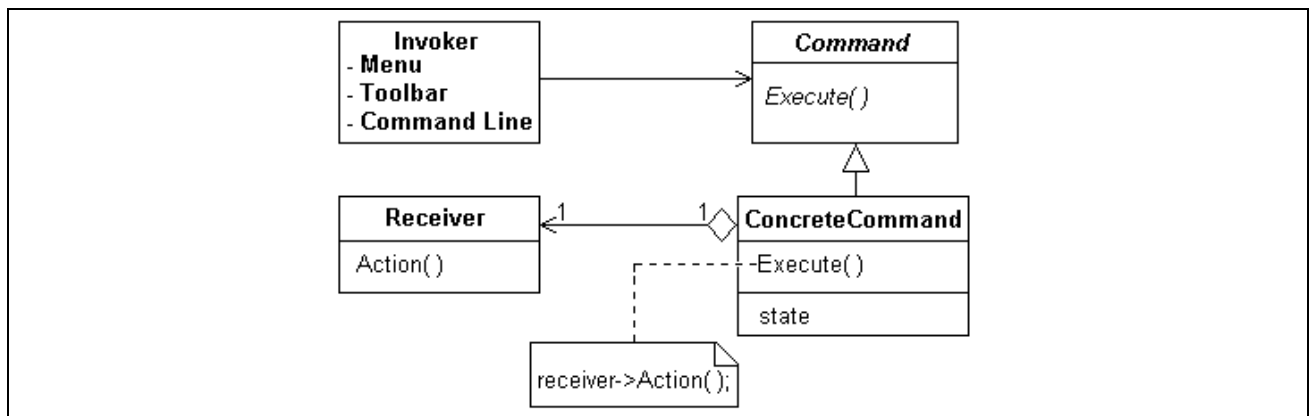



Figura III.3 – Visão geral da estrutura de dados *Command* projetada para encapsular o tratamento de comandos

Os comandos necessitam de uma lista de parâmetros, os quais deverão ser tratados da mesma forma, apesar de poderem possuir tipos diferentes. De maneira semelhante, os *labels* são definidos por um conjunto de características, cada uma delas possuindo um conjunto de atributos específicos, diferentes entre si. Nesses casos, hierarquias do tipo generalização-especialização asseguram um tratamento homogêneo entre os elementos. No caso dos *labels*, estruturas todo-parte englobam o conjunto de características para ele definidas.

III.2 – PRINCIPAIS CARACTERÍSTICAS DE IMPLEMENTAÇÃO

O primeiro passo da implementação foi realizar a integração entre o ambiente de desenvolvimento *C++ Builder* e a biblioteca gráfica *OpenGL*. Uma vez estruturado o ambiente, iniciou-se o desenvolvimento em paralelo dos subsistemas, com ênfase inicial na Interface.

III.2.1 – INTEGRAÇÃO ENTRE O BUILDER E A OPENGL

A biblioteca de funções gráficas *OpenGL* é uma ferramenta prática para a representação de objetos em três dimensões, fornecendo uma série de recursos para a visualização dos objetos representados. Entretanto, ela não é parte integrante do ambiente de desenvolvimento *Borland C++ Builder 1.0*, o que requer alguns procedimentos para a sua utilização nesse ambiente. Tornou-se necessário, portanto, encontrar um componente *OpenGL* desenvolvido para o *Builder* que fosse semelhante a um objeto do tipo *canvas*, com métodos, propriedades e funções correspondendo aos métodos, propriedades e funções da biblioteca *OpenGL*. Optou-se pelo componente criado por Alan Garny [Gar97]. Para a sua instalação foram utilizados os arquivos *OpenGL-BCB.zip*, *Glauximp.dll*, *Glauximp.lib* e *OpenGL95.exe*, com o seguinte procedimento: descompactação do *OpenGL-BCB.zip* e auto-descompactação do *OpenGL95.exe* em um diretório temporário; cópia dos arquivos com extensão ".dll" para o diretório *Bin* do *Builder*; cópia dos arquivos com extensão ".h" para o diretório *Include/GL* do *Builder*; cópia dos arquivos com extensão ".lib" para o diretório *Lib* do *Builder*; criação de um subdiretório dentro do diretório do *Builder* (sugestão: *OpenGLTools*) e cópia dos arquivos restantes para esse diretório; instalação de *Registration.obj* e *OpenGL.obj* (necessariamente nessa ordem) pelo menu *Components|Install* do *Builder*; no menu *Options|Project|Directories|Conditionals*, adição no item *include path* de *;$(BCB)\Include\OpenGLTools*; adição ao projeto em desenvolvimento dos arquivos *Registration.obj* e *glauximp.lib*. Após esses procedimentos, a *OpenGL* estará disponível como um outro componente qualquer.

III.2.2 – O AMBIENTE E O CONTROLE GERAL DO SISTEMA

Ao ativar o programa, uma janela semelhante à apresentada na figura III.4 é ativada, possuindo: um menu *pull-down* para acesso aos comandos disponíveis no modelador; barras de tarefa contendo ícones associados aos comandos do menu, a área gráfica que mostra o modelo e permite a

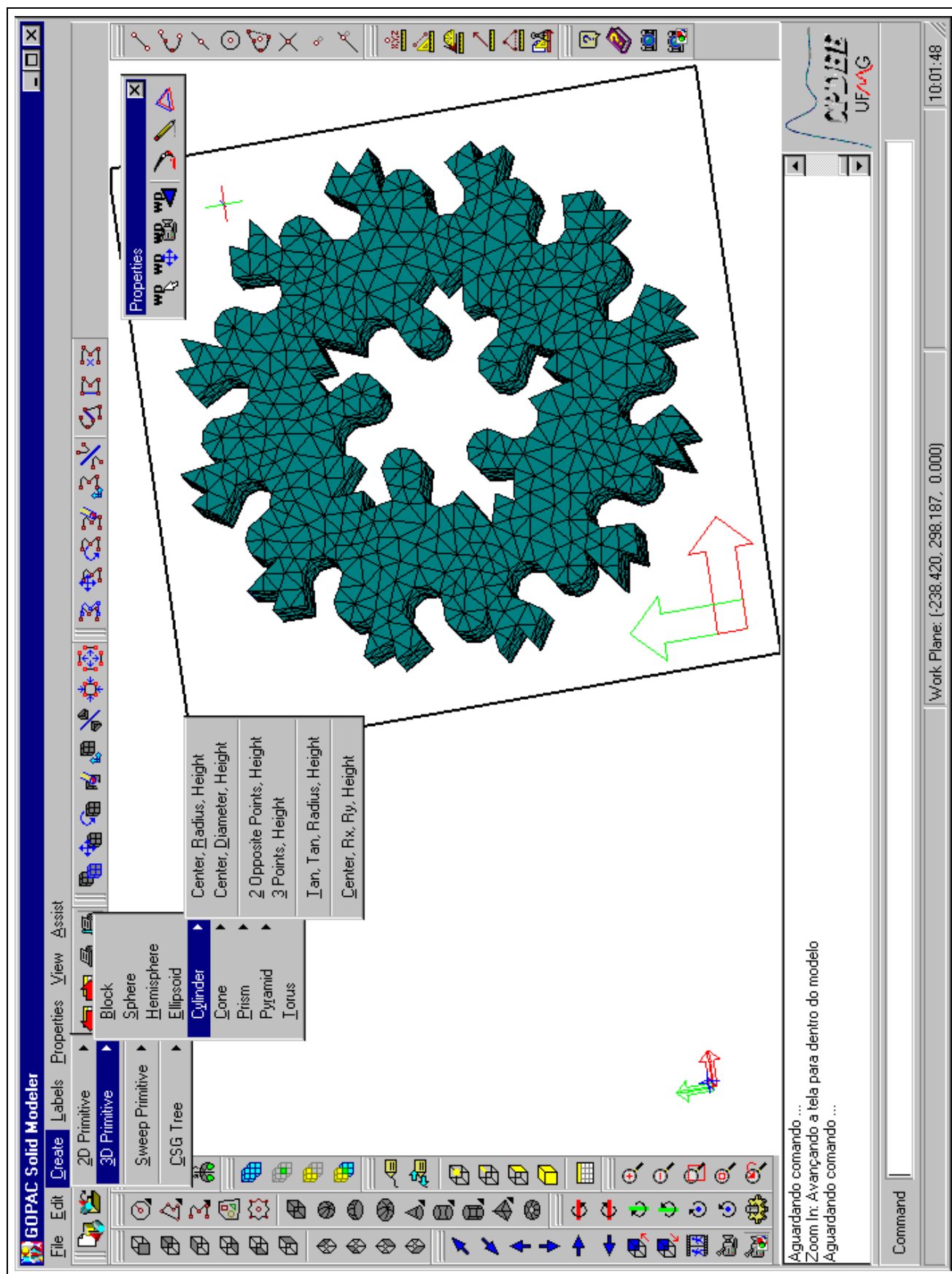



Figura III.4 – Tela principal do modelador

entrada de informações pelo *mouse*; a área de comunicação escrita da interface com o usuário, onde são colocadas as instruções de operação; a linha para entrada de comandos e dados pelo teclado; a barra de status, apresentando as coordenadas e o estado atual do sistema. A operação de todo o ambiente segue o padrão Windows. Para facilitar a localização das opções no menu *pull-down*, a figura III.5 apresenta uma representação gráfica da hierarquia de menus disponíveis no sistema. O módulo *MainInterface* é responsável pela definição de todo o ambiente; o módulo *SystemStatus* gerencia o estado geral do sistema, e o módulo *Tol* define a tolerância utilizada.

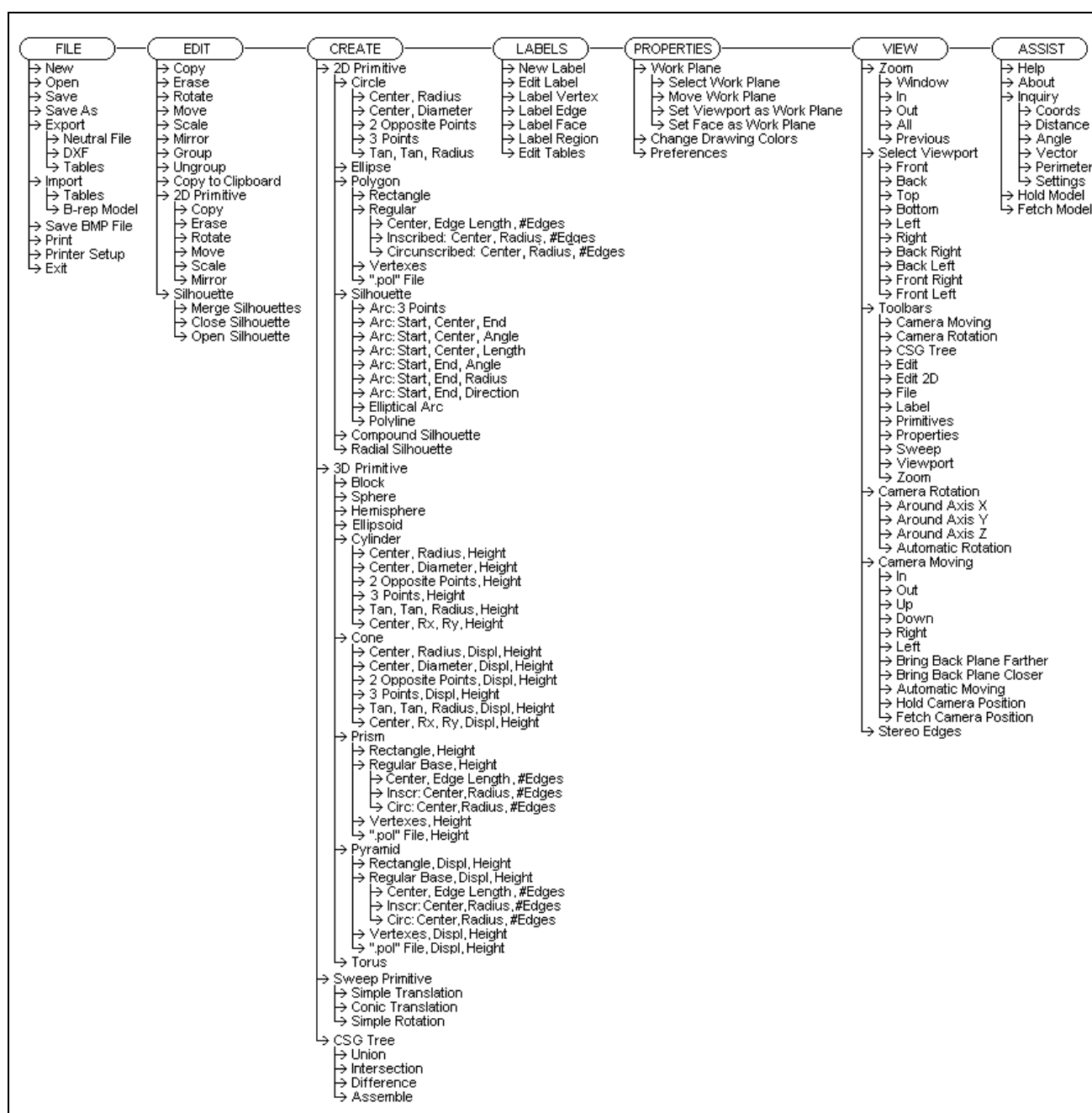



Figura III.5 – Representação gráfica da hierarquia de menus do modelador

III.2.3 – TRATAMENTO DE COMANDOS

Todo comando do menu *pull-down* possui um ícone correspondente na barra de ferramentas e um nome que pode ser fornecido pela linha de comando, conforme relacionado no Apêndice II. Qualquer uma das três formas de entrada de comando pode ser utilizada indistintamente. Para a maioria dos comandos, porém, é necessário fornecer uma seqüência de parâmetros, exigida para a sua execução. Nesta situação ocorre o seguinte problema: como garantir a correta entrada de parâmetros de um comando em um ambiente multitarefa preemptivo [Pet96], como o *Windows*?

A comunicação do *Windows* com qualquer programa aplicativo ocorre por intermédio de mensagens – como operação de teclas ou atividades do *mouse* – que são armazenadas em fila até poderem ser lidas e processadas. Parte integrante de todos os aplicativos *Windows*, o laço de mensagens recebe e processa, continuamente, mensagens que estejam na fila aguardando processamento. O aplicativo deve responder apenas às mensagens de seu interesse. Para as demais, o *Windows* geralmente fornece um processamento padrão. Não há como garantir a seqüência das mensagens na fila, nem como verificá-las.

Além dessas características peculiares do ambiente *Windows*, não há como assegurar, por parte do usuário, o fornecimento correto e na seqüência desejada de todos os parâmetros necessários em um comando. Assim sendo, foi necessário desenvolver um procedimento para gerenciar a seqüência de execução de um comando pelo modelador, por meio da construção e controle de uma máquina de estados finitos (autômato finito) que captura e interpreta seqüencialmente todos os parâmetros de cada comando. Como exemplo, o autômato gerado para o comando "*Sphere*" é apresentado no quadro III.2: ao ativar o comando, o sistema avisa que aguarda o centro da esfera; este ponto pode ser fornecido pelo teclado ou *mouse*; a seguir, o sistema solicita o raio, cujo valor pode ser diretamente fornecido pelo teclado ou calculado a partir do fornecimento de mais um ponto, pelo *mouse* ou teclado.

REPRESENTAÇÃO GRÁFICA	(diagrama)			
STATUS ATUAL	IDLE	SPHERE_CMD	SPHERE_CENTER	SPHERE_RADIUS
STATUS ESPERA	W_COMMAND	W_POINT	W_VALUE	W_COMMAND
MENSAGEM DO SISTEMA	Aguardando comando...	Sphere: Forneça o centro da esfera:	Sphere: Forneça o raio da esfera	Sphere: Gerando a esfera...
MENSAGEM DE ERRO	* Erro * Comando inválido !	* Erro * Sphere: centro ou ponto inválido	* Erro * Sphere: raio ou ponto inválido	* Erro * Sphere: geração não foi bem sucedida

Quadro III.2 – Máquina de estados finitos para o comando "*Sphere*"

Para a implementação da máquina de estados finitos foi necessário definir um *status* global para o sistema contendo, basicamente, o comando corrente, o *status* dentro do comando corrente, o tipo de entrada que está sendo aguardada (comando, ponto, valor, ângulo, vetor, seleção de um elemento, etc.) e os últimos pontos fornecidos. A cada evento ocorrido (clique do *mouse* na área gráfica, no menu ou em um ícone, pressionamento de tecla, etc.), o sistema verifica o *status* global e procura interpretá-lo dentro do que ele está aguardando receber. Desde que o tratamento seja correto, o *status* corrente é atualizado e o sistema apresenta uma nova mensagem indicando o que ele espera agora receber. Caso o tratamento não seja correto, uma mensagem de alerta avisa ao usuário sobre o erro, e outra mostra novamente o que ele espera receber. Todas essas mensagens são mostradas na área de comunicação escrita da interface com o usuário, acima da linha de comando. Um arquivo-cabeçalho contendo exclusivamente mensagens foi acoplado ao sistema, facilitando em muito a troca do idioma das mensagens: basta substituir esse arquivo por outro escrito no idioma desejado. O tratamento de comandos envolve os módulos *Comands*, *ListCom*, *Paramete*r e *Receive*r, enquanto a entrada de parâmetros é tratada pelos módulos *Keyboard* e *Mouse*.

III.2.3.1 – O FORNECIMENTO DE PARÂMETROS

Os dados necessários à execução de um comando são armazenados em um lista de parâmetros. e podem assumir diversos tipos, entre eles:

– coordenadas de um ponto, fornecidas por um clique no *mouse* ou pela digitação de seus valores. Coordenadas absolutas, relativas ou polares poderão ser fornecidas pelo teclado. É fundamental que o ambiente esteja utilizando o ponto como separador decimal para números;

– valor real fornecido para uma medida, que pode ser digitado ou calculado indiretamente pela distância entre dois pontos. Ao entrar com o parâmetro pelo *mouse*, as coordenadas de um ponto anteriormente fornecido, se existirem, serão consideradas no cálculo da distância;

– valor inteiro, fornecido somente pelo teclado;

– vetor, isto é, um valor real definindo o módulo, mais uma direção e um sentido. Pode ser digitado pelo teclado (variação em *x*, *y* e *z* ou valor e ângulo com a horizontal) ou obtido a partir do cálculo da direção e distância entre dois pontos fornecidos pelo *mouse*. Ao entrar com o parâmetro pelo *mouse*, as coordenadas de um ponto anteriormente fornecido no comando, se existirem, serão consideradas para a obtenção do vetor;

– valor angular, que pode ser digitado pelo teclado, em graus. Para a obtenção deste valor pelo *mouse*, dois pontos distintos deverão ter sido fornecidos em parâmetros anteriores, de maneira que a entrada de um novo ponto permita calcular o ângulo entre esses três pontos, tomados na ordem em que foram fornecidos;

– opção selecionada entre um conjunto de opções, que somente poderá ser fornecida digitando, pelo teclado, a letra correspondente à opção;

– elemento do desenho, que pode ser selecionado diretamente pelo *mouse* ou pela digitação de seu nome, quando esse existir.

A interpretação correta de um parâmetro dependerá do estado de espera do sistema e do evento ocorrido. Caso o comando seja cancelado, a lista de parâmetros será automaticamente esvaziada. O comando será armazenado juntamente com sua lista de parâmetros após sua execução.

III.2.4 – MANIPULAÇÃO DA ÁREA DE TRABALHO

O conceito de plano de trabalho surgiu da necessidade de obter coordenadas tridimensionais referentes ao modelo a partir das coordenadas bidimensionais da tela, capturadas pelo *mouse*. A estratégia escolhida foi o mapeamento do plano da tela em um plano limitado, localizado no espaço e definido pelo usuário – o plano de trabalho. Acessando o menu *Properties|Work Plane|Select Work Plane*, obtém-se uma interface que coleta três pontos que definem um plano retangular limitado, sobre o qual será realizado o mapeamento. Inicialmente, cogitou-se definir o plano por um ponto e sua normal, porém o resultado seria um plano infinito, necessitando ainda de três pontos para limitá-lo. Apesar de limitado, o plano de trabalho pode mover-se por todo o plano infinito no qual está contido. A partir dos pontos coletados, um objeto da classe *WPlane* é construído, sendo sua normal obtida pelo produto vetorial entre *v1=p1–p2* e *v2=p3–p2*. Note-se que a ordem de fornecimento dos pontos define a orientação da normal. Se *v2* não for perpendicular a *v1*, deve-se calcular a projeção de *v2* sobre a reta perpendicular a *v1* e coplanar a *v1* e *v2*. A partir daí, coordenadas bidimensionais de tela fornecidas pelo *mouse* são normalizadas ($0<xr<1$ e $0<yr<1$) e mapeadas em coordenadas espaciais do modelo, utilizando o seguinte procedimento: multiplicam-se os escalares *xr* e *yr* pelos vetores *v2* e *v1*, respectivamente; somam-se os vetoes *v1* e *v2* resultantes; finalmente, ao resultado de *v1+v2* soma-se o vetor que liga a origem ao ponto *p2*. A especificação do plano de trabalho envolve os módulos *WorkPlane*, responsável pela interface para entrada dos pontos de referência, e *WPlane*, que declara, gera e manipula o plano de trabalho.

III.2.5 – Recursos para Visualização

Os recursos para visualização foram implementados de forma a proporcionar ao usuário um contato tão próximo do modelo quanto possível, transmitindo-lhe a sensação de poder caminhar ao seu redor, penetrá-lo e, até mesmo, atravessá-lo. Isto tornou-se possível pela combinação dos recursos de *zoom*, de rotação, de movimentação da câmera e de remoção de superfícies escondidas.

O recurso de *Zoom* permite variar o tamanho da figura na tela, buscando uma visão do todo ou de detalhes. É formado pelos comandos *Zoom In* e *Zoom Out*, que permitem, respectivamente, aproximar e afastar o modelo.

O recurso de movimentação da câmera faz com que a tela do computador tenha função análoga à da lente de uma câmera, permitindo ao usuário deslocar-se por todo o modelo. Os comandos de avanço e retrocesso de câmera são responsáveis pelo posicionamento do plano de corte, simulando o avanço e retrocesso de uma câmera que tem liberdade de entrar e sair dos sólidos a qualquer momento. A câmera pode mover-se também para cima, para baixo e para as laterais. O modelador permite ainda mover o plano do fundo, afastando-o ou aproximando-o do ponto de observação, o que é análogo ao alcance de uma câmera e altera a região de objetos visíveis.

Os comandos de rotação permitem fazer girar o desenho, de modo a possibilitar observá-lo de todos os lados e ângulos. Estão disponíveis na forma interativa, na qual o desenho gira de um ângulo pré-definido em torno de um eixo paralelo a um dos eixos notáveis, que passa em qualquer ponto do espaço modelado, e ainda na forma automática, em que o desenho gira 360^{o} em torno de cada eixo coordenado, totalizando três voltas completas. Com a combinação de rotações interativas, pode-se obter qualquer rotação desejada em torno de um eixo qualquer.

A opção de remover ou não superfícies escondidas permite alternar as visões fio-de-arame com o ocultamento de arestas e faces não visíveis, ilustradas na figura III.6(a e b). O modelo poderá ser desenhado das seguintes maneiras: com todas as suas arestas, desprezando a presença de faces e gerando uma representação fio-de-arame (*wireframe*); somente com suas arestas visíveis, ocultando aquelas que estão encobertas por faces; com todas as faces e arestas visíveis, coloridas segundo a especificação de cores definida para cada componente, conforme apresentado a seguir. No primeiro caso, o modelo considera as faces transparentes, permitindo mostrar todas as arestas que estão escondidas atrás de faces. Já no segundo, a face é apresentada de forma opaca, mas ainda sem cor. As superfícies que estão escondidas poderão ser visualizadas utilizando os recursos de rotação ou de avanço e retrocesso de câmera, entrando e saindo dos sólidos.

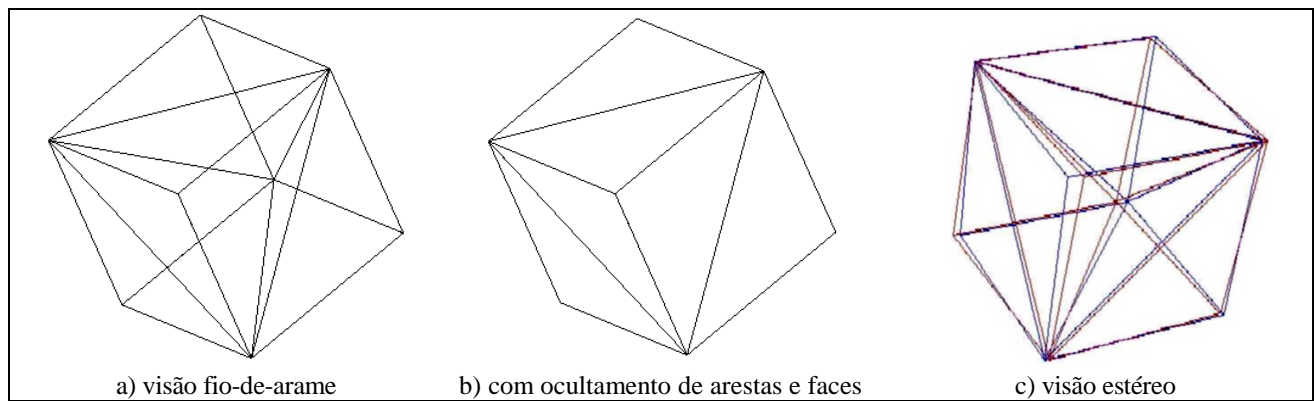



Figura III.6 – Formas alternativas de apresentação de um modelo

III.2.6 – A VISUALIZAÇÃO ESTÉREO

Além de colorir faces, remover superfícies escondidas, caminhar em volta, penetrar e afastar-se do modelo, é possível obter sua imagem tridimensional estéreo, por meio da geração de imagens anaglíficas [Wol84], como a apresentada na figura III.6c. A base da visão tridimensional reside na captura de duas imagens, pelo cérebro, a partir de pontos de vista ligeiramente separados, como ocorre na visão binocular gerada pelo olho humano. Para construir uma imagem estéreo é necessário isolar a imagem do olho esquerdo da imagem do olho direito, utilizando algum aparelho de visualização específico. Na geração de imagens anaglíficas, as imagens de um estereopar – conjunto de duas imagens-fonte – são coloridas com cores diferentes, complementares entre si. Quando as imagens são vistas através de óculos contendo lentes coloridas com essas mesmas cores, cada olho recebe apenas a informação da imagem-fonte colorida para ele, e o cérebro processa as imagens gerando a noção de profundidade. Geralmente, sob um fundo preto são geradas duas imagens-fonte, uma vermelha e outra azul, formadas pelas arestas do modelo. A imagem deverá então ser vista através de óculos contendo um filtro vermelho na lente da esquerda e um filtro azul na lente da direita. Uma vez que a mistura do azul com o vermelho resulta em preto, o filtro vermelho da lente esquerda inibirá a apresentação da cor azul e captará somente as arestas vermelhas, enquanto o filtro azul da lente direita inibirá a cor vermelha e captará apenas a imagem azul. As imagens azul e vermelha, capturadas por pontos de vista ligeiramente separados, são processadas pelo cérebro e resultam na visão tridimensional. Ao representar uma aresta, se a cor azul estiver do lado direito da cor vermelha, a aresta parecerá mais distante, atrás do plano da tela; caso contrário, se a cor vermelha estiver do lado direito da azul, a aresta parecerá mais próxima, à frente do plano da tela. Uma aresta sobre o plano da tela é representada de forma única com cor magenta, a cor vista depois que os óculos vermelho/azul fazem

convergir as linhas separadas. O maior problema das imagens anaglíficas é que elas têm que ser coloridas com tanta intensidade de azul e vermelho, que o resultado do colorido fica limitado: por isso apenas as arestas são representadas.

Desenvolvido pelo GOPAC em outra tese de doutorado [Gon00] e acoplado ao modelador, o programa *Vimesh3D* [Gon00a] permite obter uma imagem tridimensional estéreo das arestas do modelo. Usando duas câmeras, o programa cria a cena aplicando a técnica de geração de imagens anaglíficas, que constrói um estereopar azul e vermelho a ser visto através de óculos com lentes azul e vermelha. Os dados brutos do modelo são matematicamente projetados duas vezes sobre a tela do computador, cada uma com um posicionamento da câmera e uma cor (vermelho/azul) diferente. Invertendo as cores resultantes do procedimento acima explicado, o *Vimesh3D* apresenta o modelo sobre um fundo magenta, e a cor preta resulta sobre as arestas, uma vez que é vista através de ambos os filtros. A figura III.7 mostra a interface do programa, que dispõe dos seguintes recursos: modificar a espessura das linhas e o tamanho dos pontos; deslocar o modelo para o centro da tela; diminuir ou aumentar seu tamanho na tela; girar o modelo em torno de eixos x, y e z, ou girá-lo automaticamente, num efeito de animação que pode ajudar a entender melhor sua geometria.

O programa pode operar em qualquer arquivo-texto com extensão ".3D" que contenha o seguinte formato: linhas com tripletos x-y-z, separados por espaço ou tabulações, indicam os vértices; linhas consecutivas contendo vértices indicam uma polilinha a ser traçada; uma linha em branco indica o início de uma nova polilinha; a primeira linha é desprezada na leitura, servindo apenas para inclusão de comentários sobre o arquivo. Ao selecionar a opção *View|Stereo Edges*, um arquivo ".3D" com essas características é criado a partir da estrutura B-rep existente no modelador e o programa *Vimesh3D* é carregado, mostrando a visão estéreo do modelo B-rep na tela.

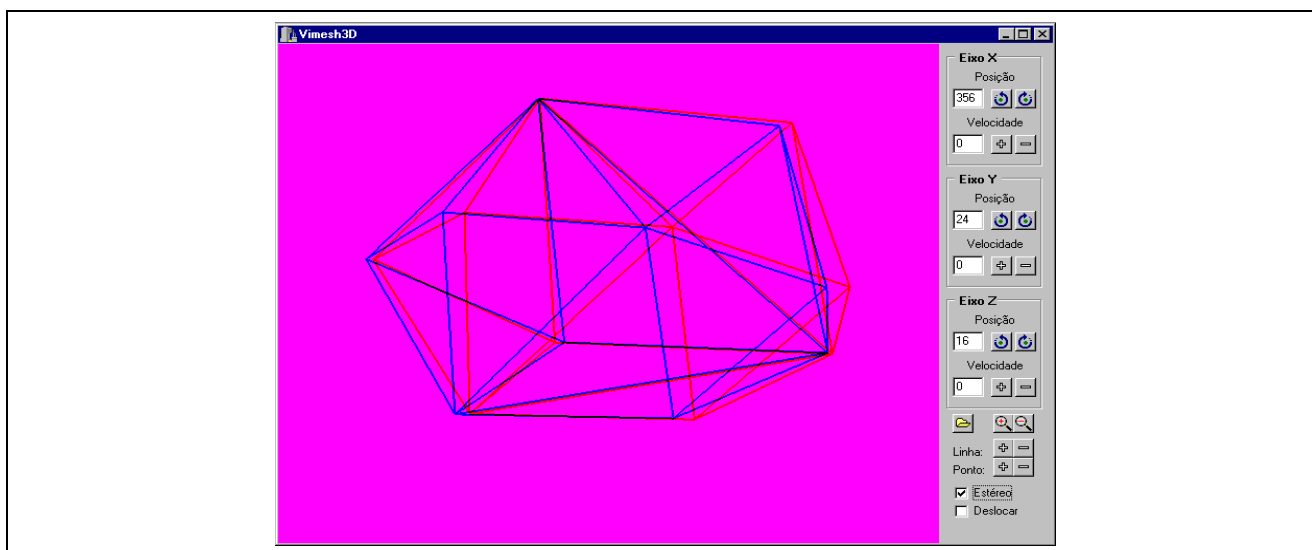



Figura III.7 – Interface do programa Vimesh3D

III.2.7 – Definição e Utilização de Cores

O modelador em desenvolvimento permite atribuir cores diferentes para cada objeto modelado. Considerando como orientação positiva o sentido anti-horário, quatro cores podem ser especificadas para cada objeto: uma para a parte frontal das faces e outra para a posterior, uma para a parte frontal das arestas e outra para a posterior. No módulo *Colors*, desenvolveu-se a classe *drawingOpenGL* para o registro das cores de cada desenho e a classe *EditColors* para a edição das cores. A classe *EditColors* define uma caixa de diálogo com o usuário para determinação das cores correntes de desenho. Esta caixa de diálogo contém cinco quadrados, cujas cores representam as cores das quatro propriedades anteriormente mencionadas, mais a cor de fundo (*background*). A cor de fundo não é armazenada com o objeto, uma vez que constitui uma propriedade geral do componente *OpenGL* – a tela gráfica – e não uma propriedade particular do objeto. Ao clicar sobre cada quadrado, aparecerá uma outra caixa de diálogo, específica para edição da cores, por meio da qual a nova cor deverá ser selecionada. Todas as operações podem ser confirmadas ou canceladas. As cores assim definidas serão utilizadas por todos os objetos criados a partir de sua especificação, até que um novo conjunto de cores seja definido. Objetos anteriormente criados não terão suas cores alteradas após uma redefinição do conjunto de cores, pois esta mudança só afetará novos objetos. Por outro lado, eles podem ser selecionados e ter suas cores alteradas. Internamente, o mecanismo funciona da seguinte forma: no momento de sua criação, cada objeto recebe uma instância da classe *drawingOpenGL*, onde são armazenadas as informações de cores. Esta instância é adicionada a uma lista de *drawingOpenGL*, que é consultada toda vez que a tela é pintada. Um procedimento semelhante gerencia a coloração de perfis bidimensionais, utilizando uma lista de objetos da classe *profileOpenGL*, que especifica apenas a cor de traçado do perfil.

III.2.8 – A Definição de *Labels*

Labels são rótulos que agrupam um conjunto de características físicas relacionadas ao modelo, a serem atribuídas aos componentes geométricos e/ou da malha de elementos finitos. Estas características envolvem tanto valores relacionados ao cálculo – tipo de material, potencial, condi-ções de contorno, valores de corrente e tensão impostas – como características de sua representação geométrica – cor, tipo e espessura de linha. Além de associar atributos físicos aos elementos geométricos e de malha, os *labels* permitem cruzar dados de geometria com os de malha e material.

Antes de proceder à criação e edição de *labels*, esquematizada na figura III.8, é necessário especificar os atributos possíveis para o modelo. Este procedimento é realizado pelo menu *Label|Edit Tables*, que apresenta na tela o formulário *Edit Tables*, composto por páginas que são selecionadas pelas suas abas. Cada página contém os controles relativos a uma determinada tabela de atributos, formados por uma lista do tipo *combo* para visualização dos atributos já criados para a tabela e dois botões, sendo um para a adição de um novo atributo e outro para a edição de um atributo selecionado, já existente. Esses botões ativam caixas de diálogo para criar e editar atributos. Após a edição das tabelas, o usuário pode criar ou editar *labels* selecionando as opções de menu *Label|New Label* e *Label|Edit Label*. Essas opções ativam uma caixa de diálogo na qual o usuário deverá digitar o nome do *label*, escolher o tipo desejado (geometria, malha ou ambos) e selecionar os atributos a partir de listas do tipo *combo*. As tabelas *Line Type* e *Potentials* não estão disponíveis no formulário de edição de tabelas por fornecerem um conjunto de atributos fixos, que não são editáveis. Exemplos dos formulários e caixas de diálogo citados serão apresentados na seção III.3. Os módulos que combinam os nomes *Append* e *Edit* com *BoundCond*, *Color*, *Current*, *IntOrder*, *Material*, *Voltage*, *Width* e *Winding* definem as interfaces e gerenciam o fornecimento de atributos, enquanto o módulo *Label* controla a interface para especificação de *labels*.

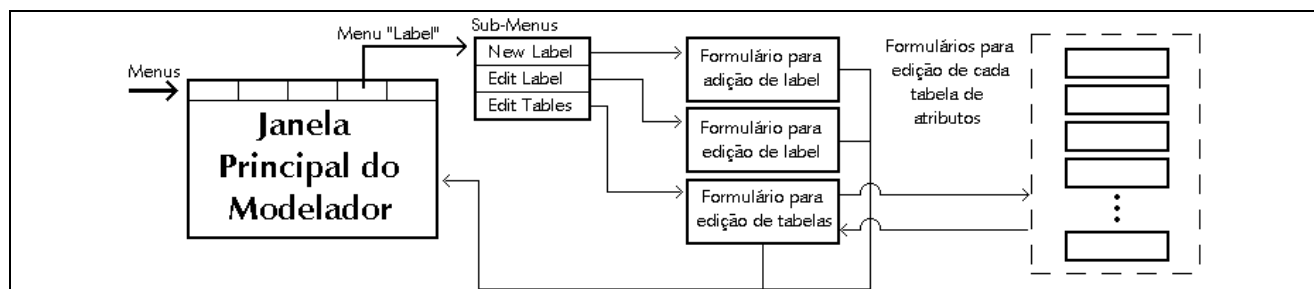



Figura III.8 – A edição de tabelas acoplada à criação e edição de *labels*

III.2.9 – APRESENTAÇÃO DE RESULTADOS

Entre os resultados gerados pelo modelador estão: a figura do modelo, apresentada continuamente na área gráfica; a base de dados neutra para intercâmbio de dados com outros aplicativos, gerada ao selecionar a opção *File|Exchange|Neutral File*; os arquivos de saída e os arquivos auxiliares, apresentados no quadro III.3 . Além de apresentar a figura do modelo, a Interface deve apenas ativar comandos e tornar disponíveis os outros resultados para o usuário: os mecanismos usados para a obtenção desses resultados são de responsabilidade exclusiva do subsistema onde foram gerados, e serão detalhados no capítulo referente a esse subsistema.

EXTENSÃO	CONTEÚDO	SUBSISTEMA
GSM	arquivo binário contendo a estrutura B-rep e todos os dados para recuperar um modelo	Núcleo
GNF	arquivo com formato neutro contendo a geometria do modelo	Núcleo
MNF	arquivo com formato neutro contendo a malha do modelo	Malha
MAT	arquivo com formato neutro contendo a descrição dos materiais do modelo	Malha
BRP	arquivo auxiliar contendo Operadores de Euler, usados na construção de um modelo B-rep	Núcleo
3D	arquivo auxiliar contendo as arestas do modelo para visualização estéreo	Núcleo
POL	arquivo auxiliar que descreve uma face planar a ser malhada pelo programa Triangle	Modelagem
ELE	arquivo auxiliar contendo os elementos triangulares gerados pelo programa Triangle	Malha
NOD	arquivo auxiliar contendo os nós dos elementos gerados pelo programa Triangle	Malha
TBL	arquivo auxiliar que permite exportar / importar características físicas já definidas	Interface

Quadro III.3 – Arquivos disponibilizados pelo sistema

III.3 – PRINCIPAIS COMPONENTES DA INTERFACE

III.3.1 – PRINCIPAIS FORMULÁRIOS E CAIXAS DE DIÁLOGO DISPONÍVEIS

As figuras mostradas a seguir apresentam as principais janelas de interação com o usuário definidas na interface do modelador. Estas janelas estão relacionadas à especificação do plano de trabalho, escolha de cores para desenho dos objetos e definição das características físicas a serem utilizadas pelos *labels* definidos no modelo. O modelador também utiliza janelas padrão do *Windows* para leitura e gravação de arquivos, configuração de impressora e impressão.

Figura III.9 – Caixas de diálogo para abertura e gravação de arquivos (todas seguem o padrão Windows)

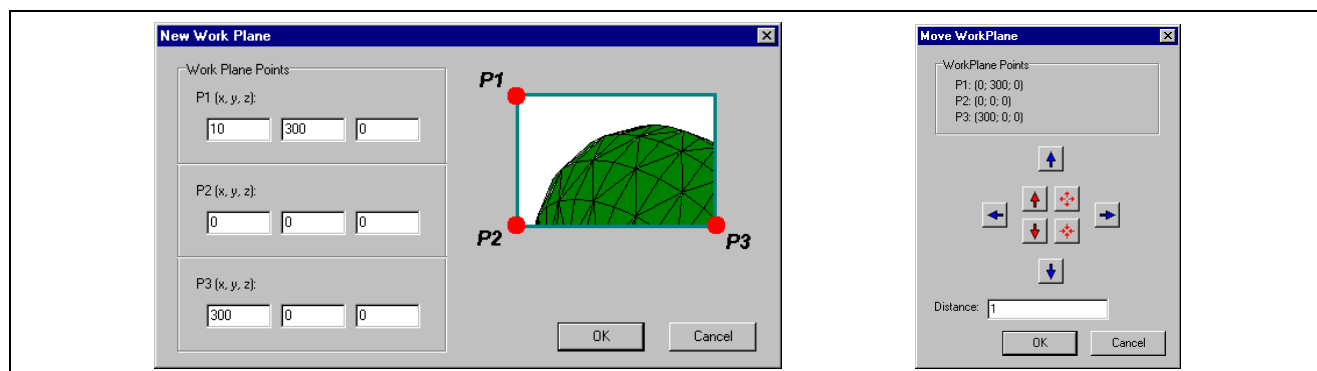



Figura III.10 – Caixas de diálogo para definição e movimentação de um plano de trabalho

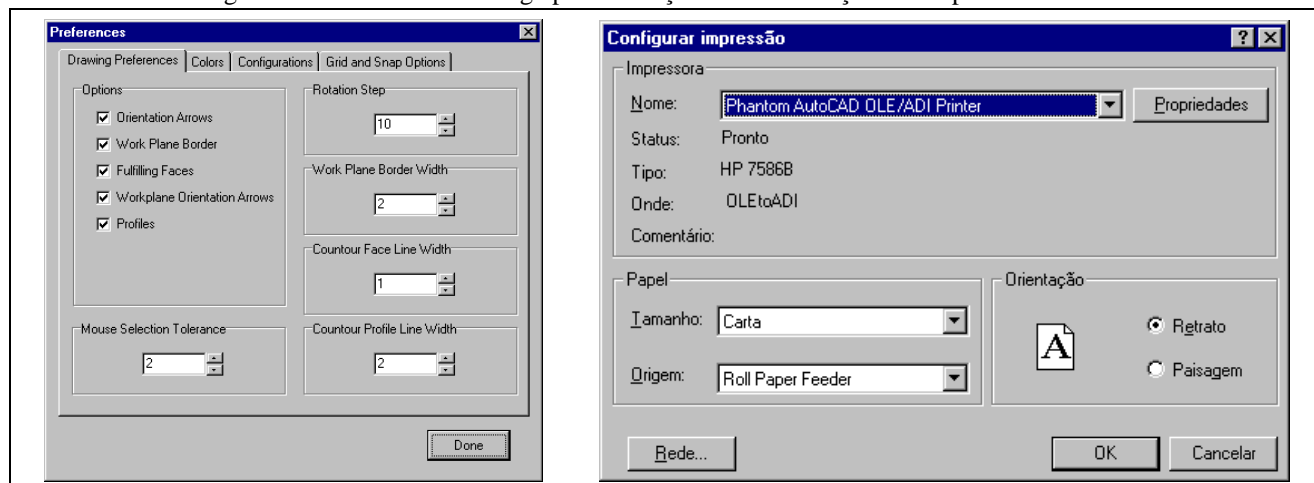



Figura III.11 – Caixas de diálogo para especificação de preferências do usuário e configuração de impressora

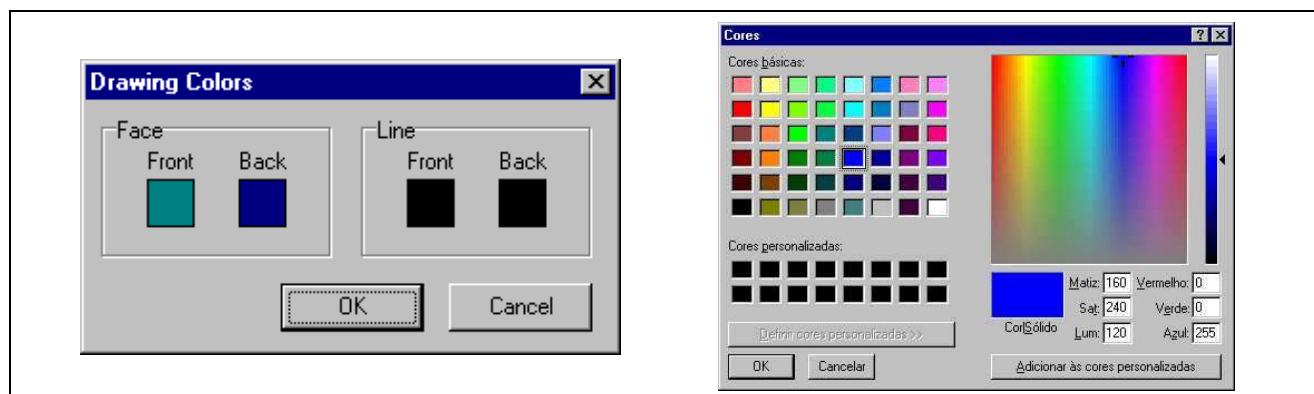



Figura III.12 – Caixas de diálogo para a definição de cores (seleção de cores segue o padrão Windows)

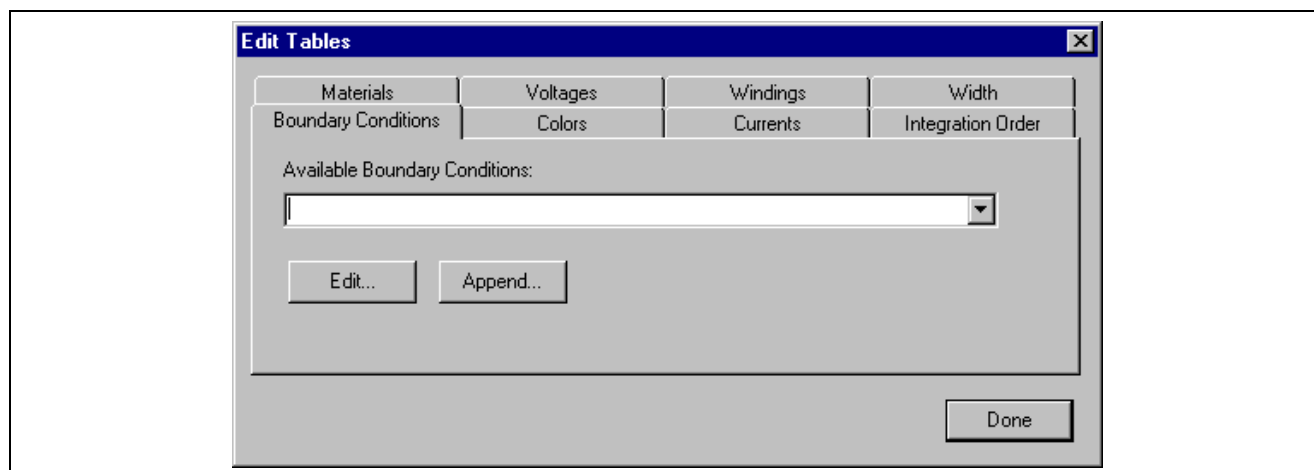



Figura III.13 – Formulário para edição de tabelas contendo características físicas

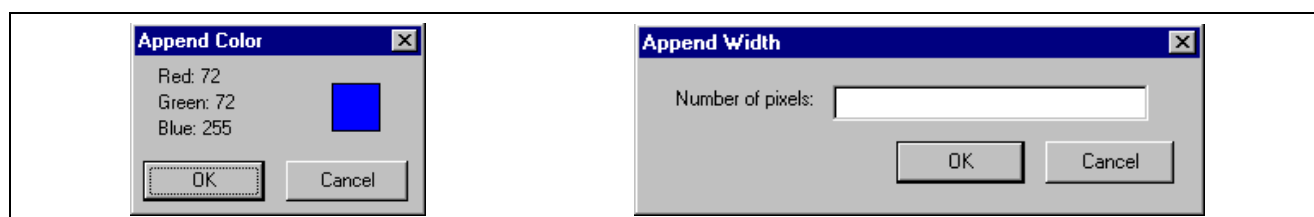



Figura III.14 – Caixas de diálogo para a definição de cor e espessura de linha (os tipos de traçado são pré-definidos)

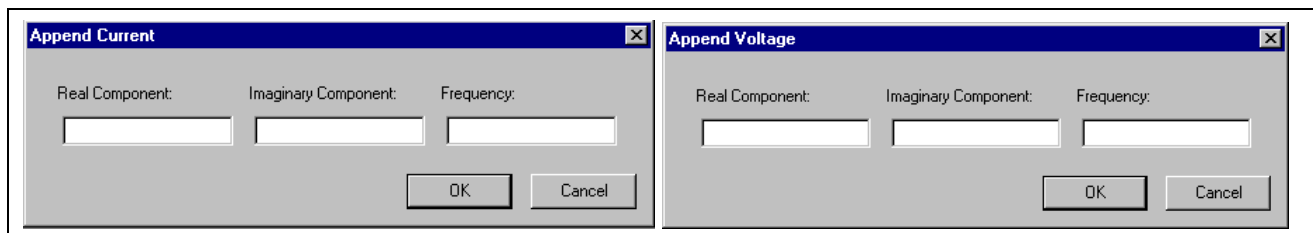



Figura III.15 – Caixas de diálogo para a definição dos atributos corrente e tensão

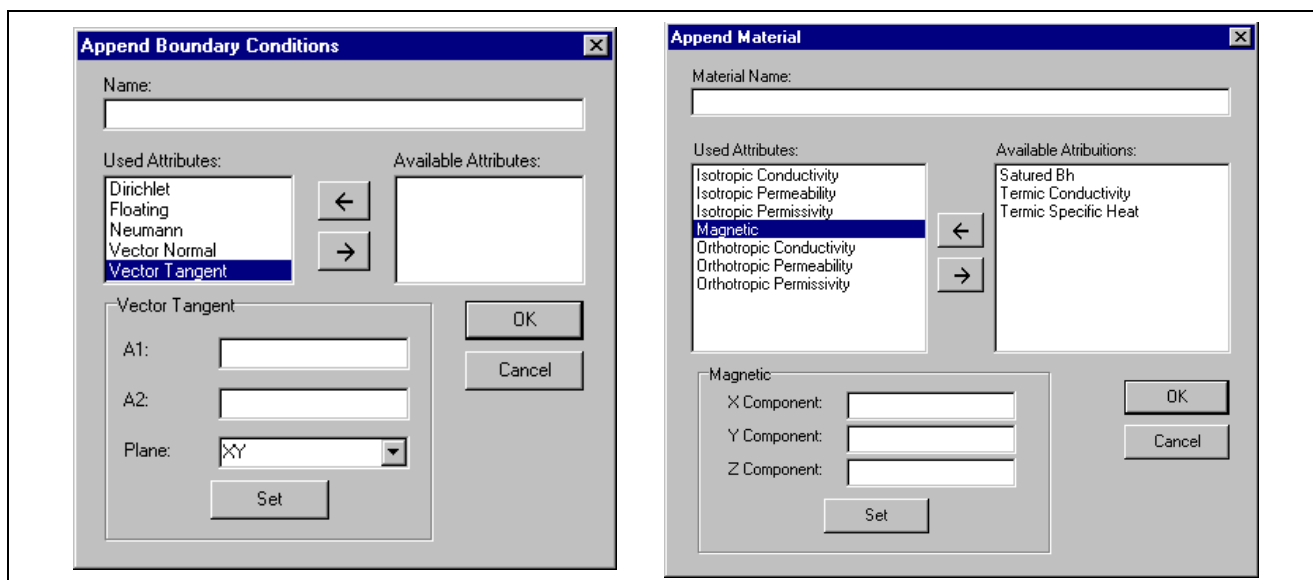



Figura III.16 – Caixas de diálogo para a definição de condições de contorno e materiais

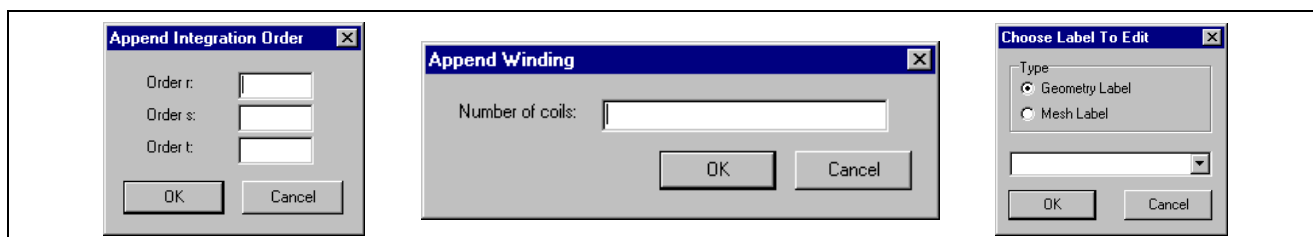



Figura III.17 – Caixas de diálogo para fornecer ordem de integração, enrolamento e tipo de *label* a ser editado

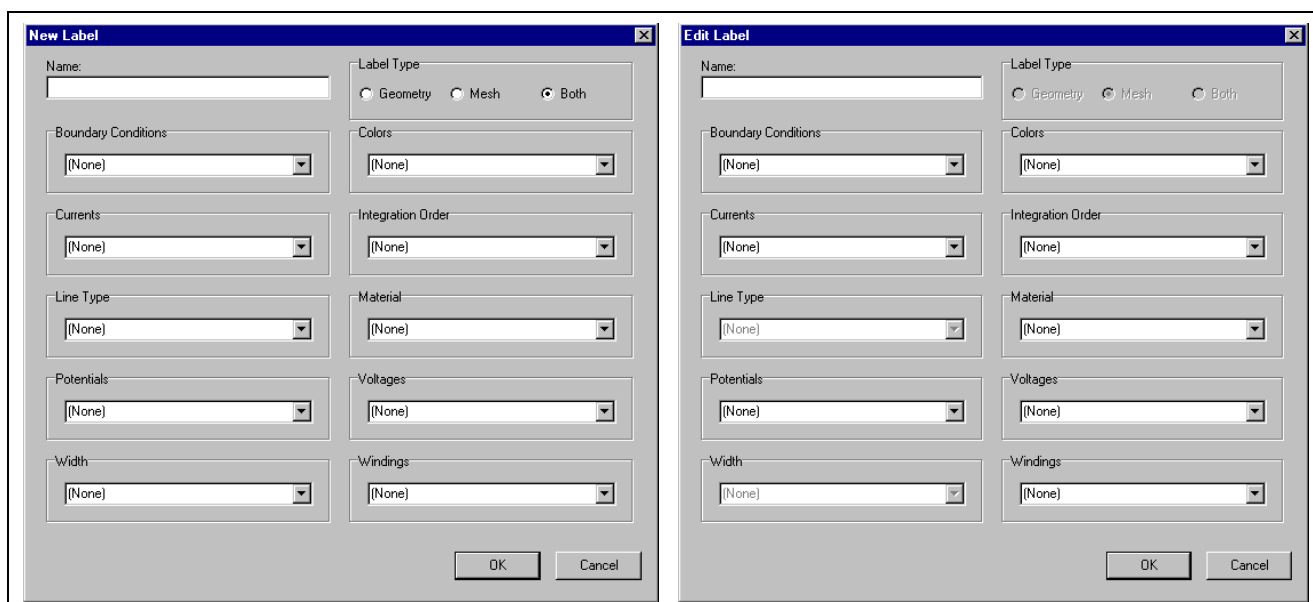



Figura III.18 – Caixas de diálogo para inclusão e edição de labels

III.3.2 – PRINCIPAIS COMANDOS DISPONÍVEIS

O quadro III.4 relaciona os principais comandos disponíveis no modelador, a serem digitados na linha de comando, selecionados pelo menu ou ativados pelo ícone existente na barra de ferramentas. A relação completa dos comandos disponíveis é apresentada no Apêndice II.

NOME DO COMANDO: DESCRIÇÃO	POSIÇÃO NO MENU	BARRA DE FERRAMENTA	ÍCONE
NEW: Reinicializa o ambiente e cria um novo modelo com nome automático	File | New	File	
OPEN: Lê um arquivo de modelo existente em disco e o carrega para a memória	File | Open	File	
SAVEAS: Grava em disco o modelo em uso com um outro nome e | ou caminho	File | Save As	File	
NEUTRALFILE: Grava em disco o modelo em uso utilizando o formato neutro	File | Export | Neutral File	File	
COPY: Copia um ou mais elementos do modelo	Edit | Copy	Edit	
ERASE: Exclui um ou mais elementos do modelo	Edit | Erase	Edit	
MOVE: Desloca um ou mais elementos do modelo	Edit | Move	Edit	
ROTATE: Rotaciona um ou mais elementos do modelo	Edit | Rotate	Edit	
MIRROR: Espelha um ou mais elementos do modelo segundo um plano qualquer	Edit | Mirror	Edit	
CIRCLECR: Gera um círculo pelo centro e raio	Create | 2DPrimitive | Circle | Center,Radius	2D Primitives	
CIRCLE3P: Gera círculo que passa pelos 3 pontos definidos	Create | 2DPrimitive | Circle | 3 Points	2D Primitives	
POLY2P: Gera retângulo por 2 pontos extremos opostos	Create | 2DPrimitive | Polygon | Rectangle	2D Primitives	
POLYVERT: Gera polígono definido pelos seus vértices (mínimo 3)	Create | 2DPrimitive | Polygon | Vertexes	2D Primitives	
SILHOUETTE: Gera silhueta com linhas, arcos circulares e elípticos	Create | 2DPrimitive | Silhouette	2D Primitives	
BLOCK: Gera um bloco de base retangular e deslocamento livre	Create | 3DPrimitive | Block	3D Primitives	
SPHERE: Gera uma esfera pelo centro e raio	Create | 3DPrimitive | Sphere	3D Primitives	
HSPHERE: Gera um hemisfério pelo centro, raio e posição	Create | 3DPrimitive | Hemisphere	3D Primitives	
ELLIPSOID: Gera um elipsóide pelo centro e raios em X, Y e Z	Create | 3DPrimitive | Ellipsoid	3D Primitives	
CONE: Gera um cone ou tronco circular ou elíptico com deslocamento livre	Create | 3DPrimitive | Cone	3D Primitives	
CYLINDER: Gera um cilindro circular ou elíptico com deslocamento livre	Create | 3DPrimitive | Cylinder	3D Primitives	
PRISM: Gera um prisma com base poliedral e deslocamento livre	Create | 3DPrimitive | Prism	3D Primitives	
PYRAMID: Gera uma pirâmide ou tronco com base poligonal e desloc. livre	Create | 3DPrimitive | Pyramid	3D Primitives	
TORUS: Gera um toro pelo centro e raios externo e interno	Create | 3DPrimitive | Torus	3D Primitives	
TRANSWEEP: Realiza a varredura translacional simples de um perfil fechado	Create | Sweep Primitive | SimpleTranslation	Sweep	
ROTSWEEP: Realiza a varredura rotacional de um perfil aberto ou fechado	Create | Sweep Primitive | SimpleRotation	Sweep	

Quadro III.4 – Principais comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas

"Deve-se estudar para saber, ler para aprender,
perguntar para entender e fazer para viver."
Gabriel Torres

IV – O SUBSISTEMA DE MODELAGEM

A modelagem de sólidos apresenta uma série de vantagens em relação às outras formas de modelagem. A principal delas é a geração de modelos menos abstratos e mais realísticos, cujo comportamento, sob uma variedade de condições de simulação, podem dizer o suficiente sobre o modo como o objeto real se comportará, tornando valioso o processo de projeto. Geralmente, um modelo complexo pode ser decomposto em um número muito menor de componentes sólidos – denominados primitivas – do que de faces, arestas ou vértices. Isso leva à conclusão de que normalmente a construção de modelos complexos é muito mais rápida quando se realiza pela composição de sólidos do que pela definição de sua fronteira.

Este capítulo é destinado ao detalhamento do processo de construção de modelos seguindo estratégia CSG – *Constructive Solid Geometry* –, utilizada no Subsistema de Modelagem. A seção IV.1 apresenta as principais características da representação CSG, fundamentais para o desenvolvimento deste subsistema. A seção IV.2 descreve a etapa de projeto, envolvendo especificação de requisitos, modelamento e soluções orientadas para objetos adotados. A seção IV.3 descreve a fundamentação teórica e implementação das primitivas e das principais operações de modelagem, entre elas as transformações geométricas, as operações booleanas e a operação de montagem.

IV.1 – O MODELAMENTO CSG

O princípio da Geometria Sólida Construtiva baseia-se na definição de sólidos complexos a partir de sólidos regulares simples – blocos, esferas, cilindros, cones, etc. – denominados primitivas, combinadas entre si por meio de um conjunto regularizado de operações booleanas – união, interseção e diferença – ou de transformações geométricas – translação, rotação e escalamento – incluídas diretamente na representação. No esquema CSG básico, exemplificado na figura IV.1a, um objeto sólido é representado como uma árvore binária, na qual cada folha é uma primitiva e cada nó interno é um operador booleano. Em diversas situações práticas, porém, como a apresentada nessa figura, um mesmo padrão de componente é utilizado diversas vezes, sendo interessante para o projetista defini-lo uma única vez e utilizá-lo sempre que precisar, aplicando sobre ele transformações geométricas

adequadas. Além de facilitar o trabalho do projetista, as transformações geométricas agilizam a construção do modelo e foram, portanto, incorporadas ao esquema CSG. A figura IV.1b mostra o mesmo objeto anterior, mas construído por uma árvore CSG contendo uma transformação geométrica: a peça é agora definida pela interseção de dois componentes topologicamente equivalentes e geometricamente idênticos, diferenciados apenas por sua posição espacial.

a) árvore CSG envolvendo apenas operações booleanas   b) árvore CSG incluindo transformação geométrica [Hui92]

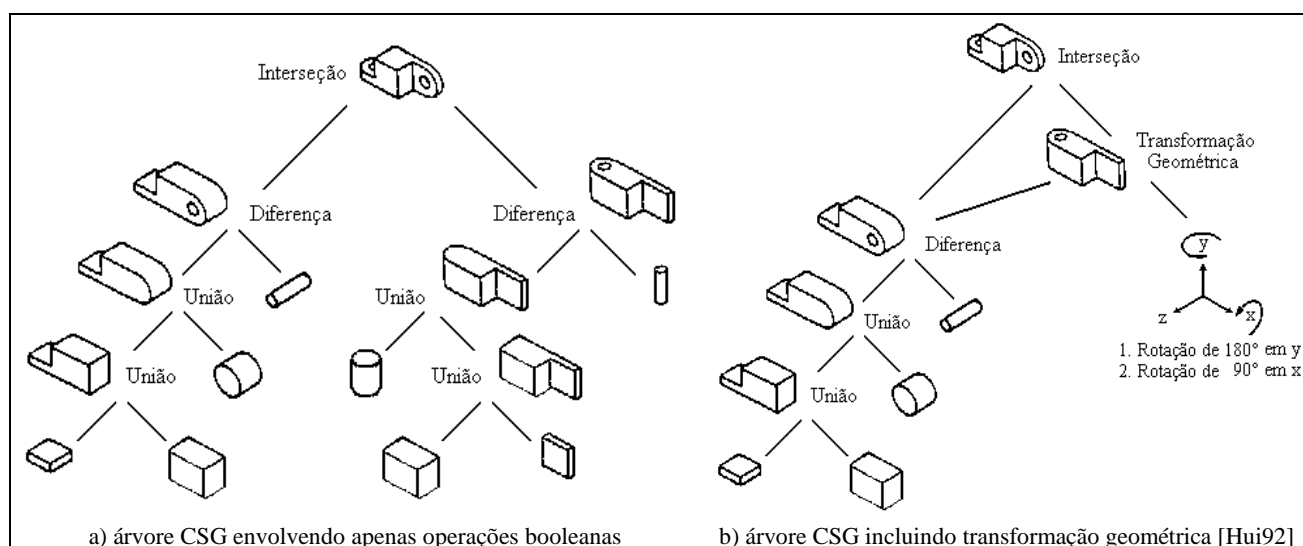



Figura IV.1 – Exemplos de árvores CSG e a melhoria obtida com a inclusão da transformação geométrica

IV.1.1 – PROPRIEDADES DA REPRESENTAÇÃO CSG

Como se constata na figura anterior, um mesmo objeto pode ser construído por árvores CSGs diferentes, o que permite concluir que o esquema CSG não possui unicidade de representação. Por outro lado, a representação CSG é não ambígua, ou seja, é completa: cada representação admitida pelo esquema corresponde a um único objeto, não deixando dúvidas quanto ao objeto representado.

A representação CSG não é uma fonte eficiente para desenhar os objetos representados, uma vez que não dispõe dos elementos gráficos utilizados para o traçado, sendo necessário avaliar a fronteira para então definir os contornos do objeto, o que constitui um processo normalmente complexo. A potência descritiva de um esquema CSG – o conjunto de objetos que podem ser por ele representados – está diretamente relacionada ao conjunto de primitivas e à generalidade das operações disponíveis.

O esquema CSG é relativamente conciso, sendo capaz de representar sólidos complexos consumindo pouco espaço de memória, apesar de normalmente serem acrescidos de informação

para agilizar e facilitar o processamento. A representação CSG possibilita executar cálculos de maneira precisa. Não obstante corretos e robustos, alguns algoritmos utilizados no esquema CSG têm pobre poder computacional que, no entanto, pode ser melhorado se forem implementados em hardware.

Dispondo de uma interface gráfica amigável, o esquema CSG permite de maneira simples a criação interativa de sólidos válidos. Partindo de primitivas regulares, qualquer representação admitida pelo esquema CSG corresponde a algum sólido válido, não sendo possível a criação de sólidos inválidos. As operações utilizadas são algebricamente fechadas para árvores CSG. O embasamento matemático rigoroso e de fácil compreensão, unido à facilidade para descrever e alterar modelos, permitiu que o esquema CSG se tornasse uma das representações de sólidos atualmente dominantes.

IV.1.2 – A Estrutura de Dados CSG Padrão

Uma **árvore** é um grafo dirigido acíclico, ou seja, é um conjunto de nós conectados por arestas que não formam ciclos, definindo uma hierarquia com as seguintes propriedades: existe um único nó, denominado raiz, que é hierarquicamente superior a todos os demais; todos os nós, com exceção do nó-raiz, possuem uma única aresta proveniente de um nó hierarquicamente superior; existe um caminho único para, a partir da raiz, atingir qualquer nó.

Uma árvore é dita ordenada quando os descendentes de cada um de seus nós seguem uma ordem, normalmente da esquerda para a direita. Um **árvore binária** é uma árvore ordenada na qual cada nó pode possuir dois descendentes: um à esquerda e outro à direita. Um nó que possui descendentes é referenciado como nó-pai de seus descendentes à direita e à esquerda e estes, por sua vez, são referenciados como nó-filho à direita e nó-filho à esquerda, respectivamente. Todo nó que possui descendentes é um nó-interno da árvore, e todo nó sem descendentes é um nó-folha. Existem vários parâmetros quantitativos associados a uma árvore, apresentados no quadro IV.1.

profundidade de um nó: é o comprimento do caminho (número de arestas percorridas) da raiz até o nó.
altura de um nó: é o comprimento do maior caminho do nó até um nó-folha.
altura da árvore: é a altura da raiz, ou seja, o comprimento do maior caminho da raiz até um nó-folha.
nível de um nó: é a altura da árvore menos a profundidade do nó.

Quadro IV.1 – Parâmetros quantitativos associados a uma árvore

Uma **árvore CSG** é uma árvore binária na qual os nós-folhas possuem primitivas, e aos nós-internos estão associadas operações booleanas ou transformações geométricas que atuam sobre seus descendentes diretos, denominados sub-árvores CSG. Cada sub-árvore representa um sólido CSG resultante das operações de combinação e transformação realizadas até aquela etapa. Obviamente, a raiz representa o objeto final. Em sua forma padrão, uma árvore CSG pode ser definida abstratamente pelas regras gramaticais BNF apresentadas no quadro IV.2. A estrutura dos nós-internos e nós-folhas da árvore CSG é apresentada no quadro IV.3.

<árvore CSG> ::= <primitiva> | <árvore CSG> <operador booleano regularizado> <árvore CSG> |
<árvore CSG> <transformação geométrica> <argumentos da transformação>
<primitiva> ::= Prisma | Pirâmide | Bloco | Cilindro | Cone | Esfera | Toro | Varredura Translacional | Varredura Rotacional
<operador booleano regularizado> ::= ∪* | ∩* | −*
<transformação geométrica> ::= translação | rotação | escalamento
<argumentos da transformação> ::= direção, distância | ângulo, eixo | fatores de escala

Quadro IV.2 – Regras gramaticais BNF para uma árvore CSG padrão

Nó-interno
- nome do sólido
- operação aplicada: booleana (∪* | ∩* | −*) ou transformação geométrica (translação | rotação) + parâmetros
- envoltório no sistema de coordenadas globais
- envoltório no sistema de coordenadas do observador
- ponteiro para o nó-filho à direita
- ponteiro para o nó-filho à esquerda

Nó-externo (folha)
- nome do sólido
- tipo do sólido (Prisma | Pirâmide | Bloco | Cilindro | Cone | Esfera | Toro | Varredura Translacional | Varredura Rotacional)
- parâmetros de definição do sólido
- envoltório no sistema de coordenadas globais
- envoltório no sistema de coordenadas do observador
- matriz de transformação do sistema de coordenadas local para o global
- matriz de transformação do sistema de coordenadas global para o local

Quadro IV.3 – Estrutura dos nós em uma árvore CSG padrão

Muitas vezes é necessário que um algoritmo percorra uma árvore CSG, ou seja, visite seus nós para obter informações ou identificar um caminho. Existem três formas distintas de percorrer ordenadamente uma árvore binária, descritas no quadro IV.4. Para determinar propriedades físicas ou para traçar um sólido representado por uma árvore CSG, por exemplo, devem-se combinar as propriedades dos nós-folhas para obter as propriedades da raiz. A estratégia de processamento utilizada é, então, o caminhamento em pós-ordem, também denominado caminhamento em profundidade.

– **Pré-ordem**: visita a raiz; visita em pré-ordem as sub-árvores à esquerda e à direita.
– **In-ordem**: visita em in-ordem a sub-árvore à esquerda; visita a raiz; visita em in-ordem a da direita.
– **Pós-ordem**: visita em pós-ordem as sub-árvores à esquerda e à direita; visita a raiz.

Quadro IV.4 – Formas de caminhamento em uma árvore binária

IV.1.3 – Variações da Estrutura CSG de Interesse para o Eletromagnetismo

A árvore CSG padrão preserva a estrutura tridimensional do objeto, fornecendo uma representação hierárquica útil para representar objetos complexos, constituindo, porém, uma representação apenas geométrica. A análise por elementos finitos necessita de informações adicionais, como propriedades físicas e topológicas. Alagar e outros pesquisadores sugerem a incorporação destas propriedades a uma representação CSG acompanhada de expressões semânticas – fornecidas junto aos nós da árvore CSG –, que garantam a exatidão das operações a serem realizadas [Ala90]. Esta estrutura recebeu de seus criadores a denominação "Árvore CSG Semântica". Segundo Alagar, com a definição cuidadosa da semântica sobre os nós nos diversos níveis de uma representação CSG, é possível obter uma certa padronização dos procedimentos para a análise por elementos finitos, atingindo uma melhoria significativa no processo como um todo. Apesar de interessantes, árvores CSG semânticas não puderam ser consideradas neste trabalho, uma vez que no modelador em desenvolvimento o esquema CSG não constitui a forma principal e permanente para armazenamento do modelo, mas apenas um meio de facilitar sua descrição pelo usuário.

Uma variação na estrutura CSG, apresentada na literatura e de grande interesse para este trabalho, é a inclusão da operação de montagem. A maioria das aplicações que necessitam de modeladores de sólidos requerem não só a representação de um componente, mas também sua manipulação em conjunto com outros componentes, sendo necessário definir uma montagem. Isso é verdade principalmente no caso do eletromagnetismo, em que se têm partes de um objeto compostas por diferentes materiais em contato direto, além de regiões de ar que envolvem esses objetos e podem conter campos eletromagnéticos. A maioria dos modeladores não dispõe de recursos para representar e manipular montagens. Essa barreira foi vencida com a inclusão do operador de montagem no modelador. A partir do formalismo para o tratamento de fronteiras internas em representações CSG, proposto por Arbab [Arb90] e detalhado na seção IV.3.6.1, foi possível definir novas operações de modelagem, que funcionam tanto sobre objetos isolados quanto sobre composições.

IV.2 – PRINCIPAIS CARACTERÍSTICAS DE PROJETO

O projeto do Subsistema de Modelagem enfatizou basicamente as operações necessárias para geração e manipulação da estrutura CSG. A especificação de requisitos procurou classificar as operações a serem desenvolvidas, o que facilitou a identificação de assuntos. A partir da especificação foram definidas as principais classes relacionadas que, agrupadas, determinaram os assuntos a serem tratados. Alguns aspectos mereceram um estudo mais detalhado, em busca de soluções orientadas para objetos, a serem adotadas pela implementação.

IV.2.1 – REQUISITOS RELACIONADOS À MODELAGEM

O Subsistema de Modelagem é responsável pela manipulação da forma dos componentes do modelo. Sua principal função é traduzir as descrições de objetos e operações obtidas da Interface em comandos para a criação e edição das representações internas, de forma a: garantir a geração de primitivas 3D válidas; possibilitar a edição correta de primitivas 3D; realizar transformações geométricas sobre primitivas e demais componentes do modelo; obter sólidos mais complexos, pela combinação de primitivas por meio de transformações geométricas, operadores booleanos e de montagem; fornecer operações de suporte à visualização; prover operações de apoio e responder a questões geométricas e topológicas.

Cabe também ao Subsistema de Modelagem executar as operações de modelagem solicitadas pelo usuário por intermédio da Interface. Muitas dessas operações implicam o fornecimento de parâmetros, envolvendo funções de identificação e posicionamento fornecidas pela Interface. Caso ocorra algum erro durante uma operação de modelagem, o usuário deverá ser notificado. As operações desenvolvidas neste subsistema podem ser classificadas de acordo com sua função:

– operações de definição de primitivas, usadas para verificar a consistência dos parâmetros fornecidos e calcular os dados necessários para a geração da estrutura de dados correspondente à primitiva. Uma vez recebidos os parâmetros de definição da primitiva, o Subsistema de Modelagem deve verificar se eles permitem a criação de uma primitiva válida, realizar os cálculos necessários para especificar as operações de construção da primitiva e solicitar ao Subsistema de Representação a criação da estrutura de dados correspondente;

– operações de manipulação, usadas para a construção e manipulação da forma de sólidos, alterando os atributos de uma composição anteriormente definida ou criando novas composições. Podem ser combinadas entre si, dando origem a novas operações. As operações básicas de manipulação no modelador são: as transformações geométricas, com opção para duplicação do componente, envolvendo a translação, a rotação, o escalamento – que inclui o alongamento ou encolhimento – e possivelmente o espelhamento; as operações booleanas de união, interseção e diferença; a operação adicional de montagem, sendo que todas elas necessitam de uma série de rotinas auxiliares para sua execução;

– operações geométricas de apoio, funções que utilizam informações geométricas do modelo para gerar outras informações geométricas a serem usadas na manipulação ou definição de sólidos. Percorrem a estrutura de dados acessando e manipulando apenas as informações geométricas, que são usadas para responder a questões geométricas ou verificar a validade dos objetos após a execução de uma operação. Normalmente compreendem as rotinas de interseção entre retas, curvas, planos e superfícies, projeções de curvas ou superfícies sobre um plano e classificação ou verificação de pertinência entre entidades geométricas;

– operações de avaliação da fronteira, usadas na conversão entre as representações CSG e B-rep. Incluem as operações geométricas de apoio e as operações de Euler, utilizadas na construção da representação da fronteira do modelo, atuando sobre a conectividade entre faces, arestas e vértices, e garantindo sua consistência topológica;

– operações de agrupamento, que permitem o tratamento em grupo de componentes anteriormente definidos, sendo usadas no modelamento de elementos mais complexos. O uso de operadores de manipulação sobre qualquer das entidades do grupo tem efeito sobre todas as demais. Atua sobre a estrutura de dados, associando a um grupo entidades já definidas;

– operações de apoio, usadas como ferramentas auxiliares para a obtenção de informações sobre os elementos do modelo e como auxílio na definição de parâmetros para as demais operações. Não estão relacionadas diretamente com a criação e manipulação do modelo.

– operações de apoio à visualização, usadas na obtenção de vistas do objeto a partir de diferentes posições, com possibilidade de ampliação / redução da área de trabalho sobre o modelo e remoção de linhas e superfícies escondidas. Podem ainda incluir rotinas de visualização em corte, permitindo obter vistas do modelo através de corte definido por um plano ou por uma combinação de planos.

IV.2.2 – O MODELAMENTO OBTIDO

A especificação de requisitos do Subsistema de Modelagem permitiu identificar as principais classes e as relações de dependência entre elas, ilustradas na figura IV.2. Uma análise um pouco mais detalhada facultou o acesso aos atributos e métodos essenciais de cada classe. Pelo resultado obtido, esquematizado no quadro IV.5, foi possível identificar os assuntos tratados pelo Subsistema de Modelagem bem como relacioná-los entre si e com o demais assuntos do sistema. O resultado, apresentado na figura IV.3, é explicado a seguir.

O assunto Elementos Geométricos engloba definição, instanciamento e manipulação dos elementos geométricos (ângulo, ponto, plano, plano limitado, reta, segmento de reta e vetor contendo módulo, direção e sentido). O elemento ponto é utilizado por todo o sistema, enquanto os demais são utilizados na definição de primitivas 2D e 3D, na representação CSG e na definição da representação B-rep pelo Subsistema de Representação.

De uma forma geral, uma Primitiva 3D pode ser definida por uma Primitiva 2D mais uma operação de varredura, que pode ser translacional simples ou cônica, ou rotacional de um perfil aberto ou fechado. As operações de varredura incluem as classes *SweepOp*, *RotSweep* e *TranSweep*. Para definir Primitivas 2D são usados arcos, polilinhas ou uma combinação de ambos, denominada silhueta. Uma Primitiva 3D definida por varredura instancia um objeto da classe *SweepPrim*, composto por uma Primitiva 2D e uma operação de varredura. As Primitivas 3D – cone, cilindro, prisma, pirâmide, esfera, elipsóide, hemisfério e toro – podem ser geradas pela definição de seus parâmetros específicos, sendo então passadas para o Subsistema de Representação onde, pelos Operadores de Euler, será gerada a representação B-rep correspondente.

Além das primitivas, a representação CSG abrange os assuntos: Operações Booleanas, que instanciam objetos do tipo *BoolOp* para realizar as operações de união, interseção ou diferença de componentes CSG; Transformações Geométricas, que instanciam objetos do tipo *GeoTransfOp* para realizar transformações de rotação, translação, escalamento ou reflexão; Operação de Montagem, que instancia objetos do tipo *AssembOp* para realizar o acoplamento de componentes, definindo fronteiras entre eles. As classes bases *CSGSolid*, *CSGNode*, *CSGComponent*, *CSGSubTree* e *CSGLeaf* definem o modelo CSG. Associadas à representação CSG, operações geométricas de apoio auxiliam na avaliação da fronteira, a ser realizada pelo Subsistema de Representação. Uma margem de tolerância é utilizada para realizar a aproximação de curvas por segmentos de retas e de superfícies por facetas planares. Matrizes e vetores, estruturas de dados auxiliares, são utilizados para armazenar coordenadas, distâncias e outros valores relacionados às primitivas e a operações de modelagem.

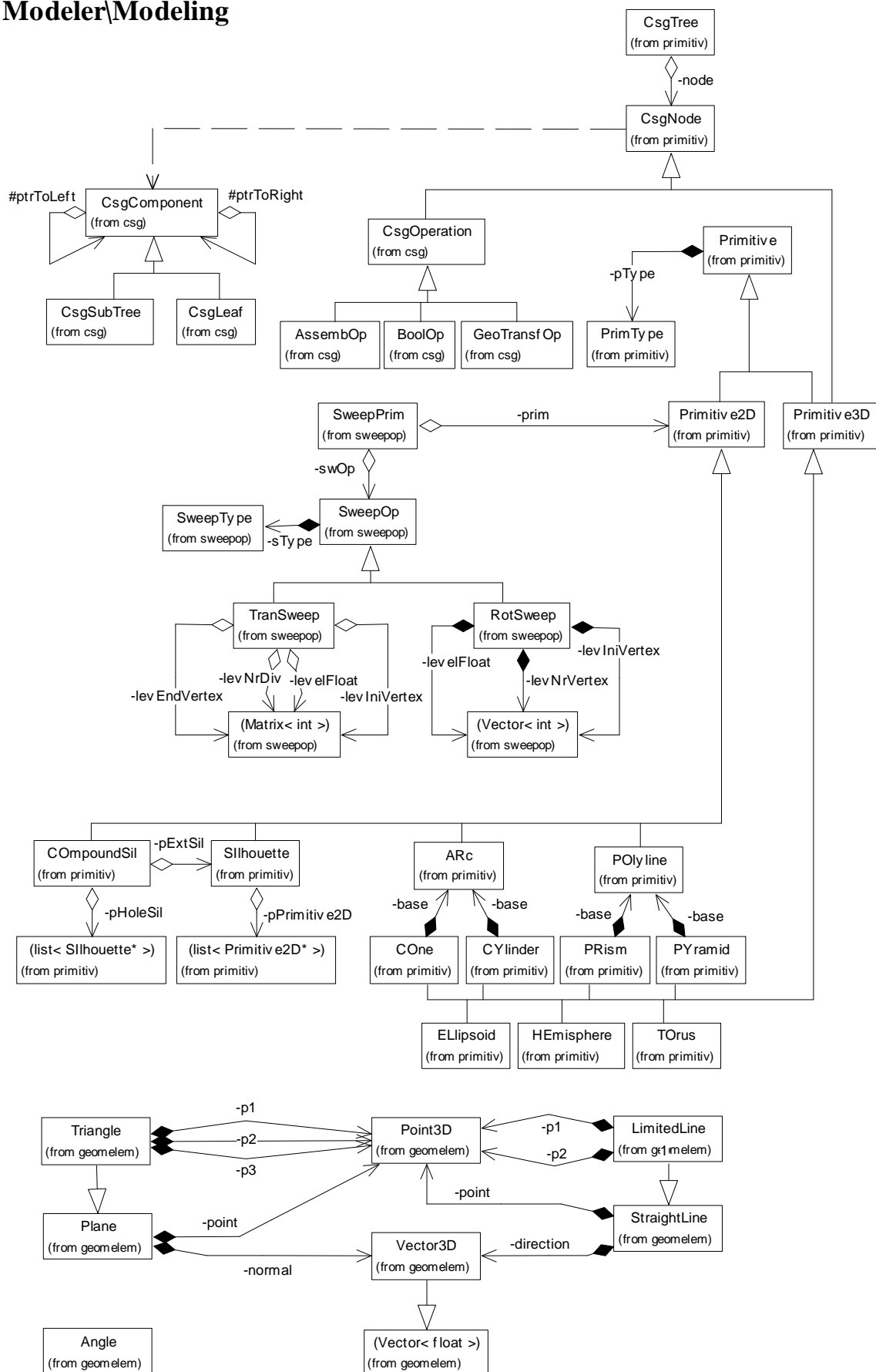



Figura IV.2 – Diagrama das principais classes e relações de dependência entre elas

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
CsgTree: classe base para a árvore CSG	
Identificador da raiz da árvore CSG	Englobar todos os nós para um modelo CSG
CsgNode: um nó da árvore CSG, que pode ser um componente ou uma operação	
Tipo do nó	Obter tipo, homogeneizar tratamento do nó
CsgComponent: um componente da árvore CSG, que define um nó interno ou um nó folha	
Identificador do componente (único para a classe), especificação do próprio nó e dos filhos à direita e à esquerda	Homogeneizar o tratamento de componentes
CsgSubTree: uma sub-árvore a árvore CSG, filha de CsgComponent, que pode conter descendentes	
Nenhum	Mostrar sua seqüência de criação, garantir o tratamento recursivo de sub-árvores
CsgLeaf: componente da árvore CSG, filha de CSGComponent, que não pode conter descendentes	
Nenhum	Terminar o tratamento recursivo de sub-árvores
Primitive: classe abstrata que engloba todas as primitivas 2D e 3D	
Tipo da primitiva	Obter tipo, homogeneizar tratamento de primitivas
Primitive2D: filha de Primitive, engloba todas as primitivas 2D	
Lista de vértices, *flag* indicando se fechada, plano de definição	Homogeneizar tratamento de primitivas 2D, desenhar, obter vértices, calcular vértices intermediários, baricentro, perímetro
Arc: primitiva para definição de arcos, círculos ou elipses	
Centro, raioX, raioY, ângulos inicial e final, tolerância	Desenhar, obter vértices, calcular vértices intermediários, baricentro, perímetro
Polyline: primitiva para definição de polígonos e polilinhas	
Nenhum, apenas herda atributos de Primitiv2D	Desenhar, obter primeiro e último vértices, calcular vértices intermediários, baricentro, perímetro
Silhouette: primitiva para definição de silhuetas envolvendo polilinhas e arcos	
Lista de primitivas 2D	Desenhar, obter vértices, calcular vértices intermediários, baricentro, perímetro
CompoundSil: primitiva para definição de perfis compostos por mais de uma borda	
Silhueta externa, lista de silhuetas internas (que definem buracos)	Gerenciar montagem e consistência de perfis compostos, delegando demais serviços à silhueta
Primitive3D: filha de Primitive, engloba todas as primitivas 3D	
Nenhum	Homogeneizar tratamento de primitivas 3D
Cone: primitiva que define um cone circular ou elíptico	
Arco da base (círculo ou elipse), ápice da pirâmide, altura real (para tronco)	Desenhar, identificar-se como nó da árvore CSG
Cylinder: primitiva que define um cilindro circular ou elíptico	
Arco da base, vetor deslocamento	Desenhar, identificar-se como nó da árvore CSG
Ellipsoid: primitiva que define uma esfera ou uma elipsóide	
Centro, raioX, raioY, raioZ,	Desenhar, identificar-se como nó da árvore CSG
Hemisphere: primitiva que define um hemisfério	
Centro, raio, lado em que está definido (um dos lados do plano de trabalho)	Desenhar, identificar-se como nó da árvore CSG
Prism: primitiva que define um prisma	
Polilinha da base, vetor deslocamento	Desenhar, identificar-se como nó da árvore CSG
Pyramid: primitiva que define uma pirâmide – válida também para definir bloco	
Polilinha da base, ápice da pirâmide, altura real (para tronco)	Desenhar, identificar-se como nó da árvore CSG
Torus: primitiva que define um toro circular ou elíptico	
Centro, raioX interno, raioX externo, raioY, raioZ	Desenhar, identificar-se como nó da árvore CSG

Quadro IV.5 – Principais características das classes relacionadas (início)

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
SweepPrim: primitiva definida por varredura	
Primitiva 2D, operação de varredura	Homogeneizar o tratamento da varredura
SweepOp: filha de SweepPrim, engloba todas as operações de varredura	
Tipo da varredura, perfil redefinido, coordenadas obtidas, segmento padrão utilizado na geração	Obter as coordenadas resultantes da operação de varredura aplicada
RotSweep: Filha de SweepOp, realiza a varredura rotacional	
Eixo de varredura, ângulos inicial e final, número de setores	Obter as coordenadas resultantes da varredura rotacional, gerar operadores de Euler para construir a representação B-rep
TranSweep: Filha de SweepOp, realiza a varredura translacional	
Vetor deslocamento em x, y e z; altura real (para tronco na varredura translacional cônica)	Obter as coordenadas resultantes da varredura translacional, gerar operadores de Euler para construir a representação B-rep
CsgOperation: filho de CsgNode, engloba todas as operações envolvidas na estrutura CSG	
Tipo da operação CSG	Obter tipo, homogeneizar tratamento de operações
GeoTransfOp: filho de CsgOperation, define uma matriz em coordenadas homogêneas para transformações geométricas	
Matriz em coordenadas homogêneas	Gerar identidade, gerar matrizes de transformação, identificar-se como nó da árvore CSG
BoolOp: filho de CsgOperation, define uma operação booleana	
Tipo da operação booleana (união, interseção, diferença)	Obter tipo, identificar-se como nó da árvore CSG
AssembOp: filho de CsgOperation, define uma operação de montagem	
Identificadores das regiões a serem montadas	Verificar regiões, identificar-se como nó da árvore CSG
Point3D: um ponto no espaço tridimensional	
Coordenadas x, y e z	Converter coordenadas cilíndricas e esféricas, calcular distância, ponto médio, circuncentro, verificar colinearidade, realizar operações diversas
Angle: define ângulos em graus	
Valor do ângulo	Converter graus para radianos, calcular seno, coseno, tangente, realizar operações diversas (soma, divisão, comparações, etc.)
Plane: define um plano de dimensões infinitas	
Vetor normal, um ponto qualquer do plano	Verificar paralelismo com reta e com plano, calcular interseção com reta e com plano, ângulo entre dois planos, distância de um ponto ao plano
Triangle: filho de Plane, define uma superfície plana triangular limitada por três pontos não colineares	
Três pontos não colineares	Verificar se contém ponto, segmento de reta, outro triângulo e demais funções de Plane
StraightLine: define uma reta infinita	
Vetor diretor, um ponto qualquer da reta	Verificar coplanaridade e paralelismo com reta, calcular interseção com reta, ângulo entre duas retas, distância de um ponto à reta
LimitedLine: define um segmento de reta	
Dois pontos extremos do segmento	Verificar se contém ponto, outro segmento e demais funções de StraightLine
Vector3D: define um vetor com módulo, direção e sentido	
Vetor contendo deslocamentos em x, y e z	Verificar paralelismo, calcular módulo, vetor unitário, produto escalar, produto vetorial, soma, subtração, divisão por escalar

Quadro IV.5 – Principais características das classes relacionadas (continuação)

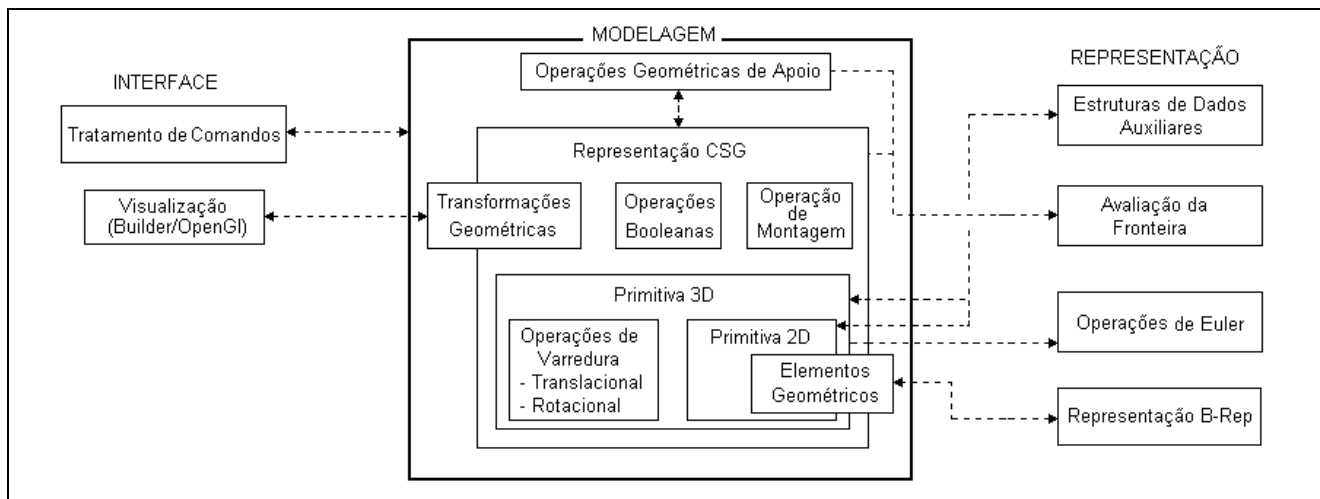



Figura IV.3 – Identificação de assuntos relacionados ao Subsistema de Modelagem

IV.2.3 – SOLUÇÕES ORIENTADAS PARA OBJETOS

Além de dispor das primitivas normalmente utilizadas em modeladores CSG, este modelador permite utilizar como primitiva qualquer sólido obtido por varredura translacional ou rotacional de um perfil, ampliando o conjunto básico de primitivas disponível. Dentro do projeto orientado para objetos, definido para o Subsistema de Modelagem, uma das principais classes é a *Primitive3D*, esquematizada na figura IV.4, que representa as primitivas sólidas utilizadas na construção de árvores CSG. Seu modelo orientado para objetos representa uma hierarquia de estruturas envolvendo relações de dependência do tipo:

– Generalização ⇔ Especialização, em que as classes podem possuir, além de suas próprias características, as características gerais referentes à classe de nível superior, implementando o conceito de herança entre classes. Na figura IV.4, isto ocorre entre a classe *Primitive2D* e suas especializações *Arc*, *Polyline*, *Silhouette* e *CompoundSil*; entre a classe *Sweep Op.* e suas especializações *Simple Transl.*, *Conic Transl.*, *Opened Rotat.* e *Closed Rotat.* e entre a classe *Primitive3D* e suas especializações *Cylinder, Cone, Ellipsoid, Hemisphere, Prism, Pyramid* e *Torus*;

– Todo⇔Parte, nas quais a entidade que está no nível inferior faz parte da entidade de nível superior. Na figura IV.4, isso ocorre entre a classe *SweepPrim* (o "todo") e as classes *Primitive2D* e *Sweep Op.* (as "partes"). Para cada instância de *SweepPrim*, existe uma instância de suas classes filhas.

Também é possível notar a ocorrência de uma estrutura múltipla na definição da classe *Primitive3D*, indicando que ela é obtida por um processo de varredura, e que todas as primitivas sólidas pré-definidas (*Cylinder*, *Cone*, *Ellipsoid*, etc.) estão baseadas em operações de varredura.

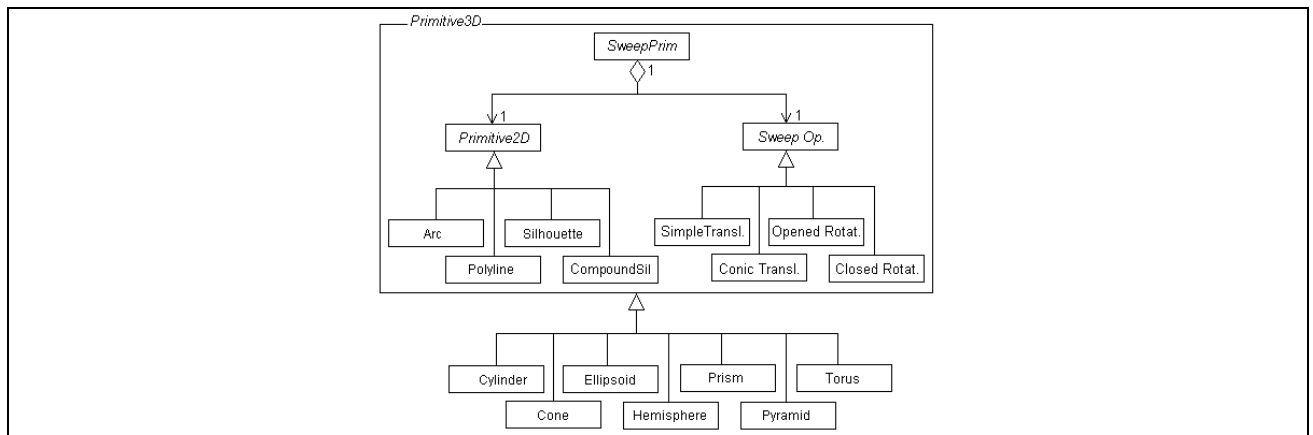



Figura IV.4 – Relações de dependência na estrutura *Primitive3D*

Outro modelo orientado para objetos, importante no Subsistema de Modelagem, é o da representação CSG, apresentado na figura IV.5, que descreve a árvore binária *CSGTree* utilizada, composta por *CSGNode*s, sendo os nós-folhas primitivas sólidas (da classe *Primitive3D*) e os nós-internos uma operação booleana (*Boolean Op*.), uma transformação geométrica (*Geometric Op*.) ou uma montagem (*Assemble Op*.). A classe *CSGModel* pode possuir várias *CSGTree*s, ou seja, várias árvores, cada uma representando um componente desconexo do modelo, formando assim uma "floresta". Isso permite que um modelo CSG seja formado por uma composição de árvores CSG definidas.

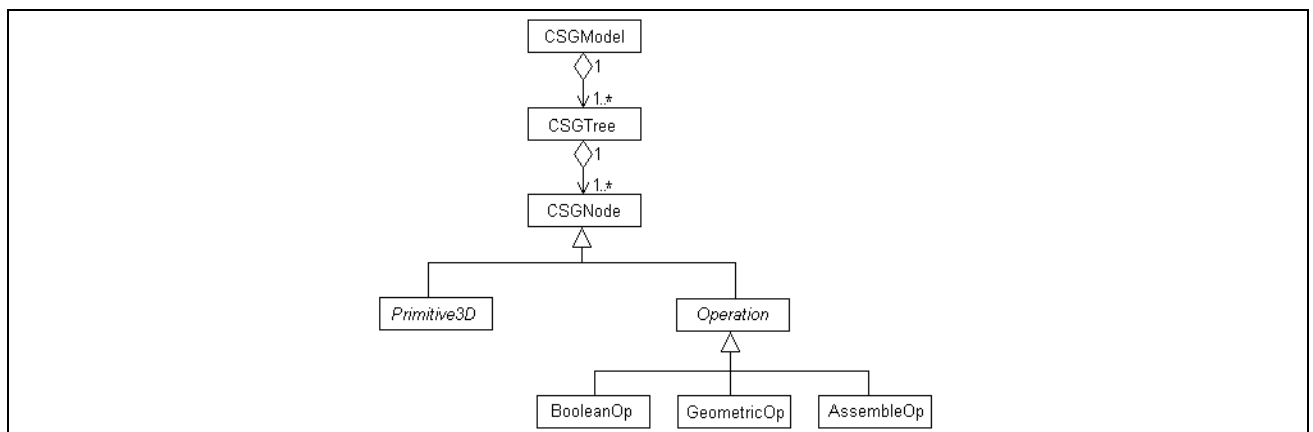



Figura IV.5 – Relações de dependência na estrutura *CSGModel*

Modelos CSG são normalmente representados por árvores CSG, com primitivas nos nós-folhas e operações booleanas ou transformações geométricas nos nós-internos. Para evitar redundância na definição de sólidos, sub-árvores CSG podem ser usadas em substituição a primitivas. Se o usuário pode tratar primitivas e sub-árvores CSG de forma idêntica, é desejável que o código que manipula estas classes também o faça de maneira uniforme. Isso foi conseguido utilizando o padrão *Composite*, uma estrutura arborescente que representa hierarquias todo-parte,

como mostrado na figura IV.6. A idéia principal deste padrão é a classe abstrata *CSGComponent*, que representa tanto primitivas quanto sub-árvores CSG. Essa classe declara não só operações das quais todas as sub-árvores CSG compartilham, tais como as que são usadas para acessar e manipular seus descendentes, mas também operações específicas para primitivas. Como as primitivas não possuem descendentes, nenhuma destas subclasses implementa operações relacionadas a dependentes.

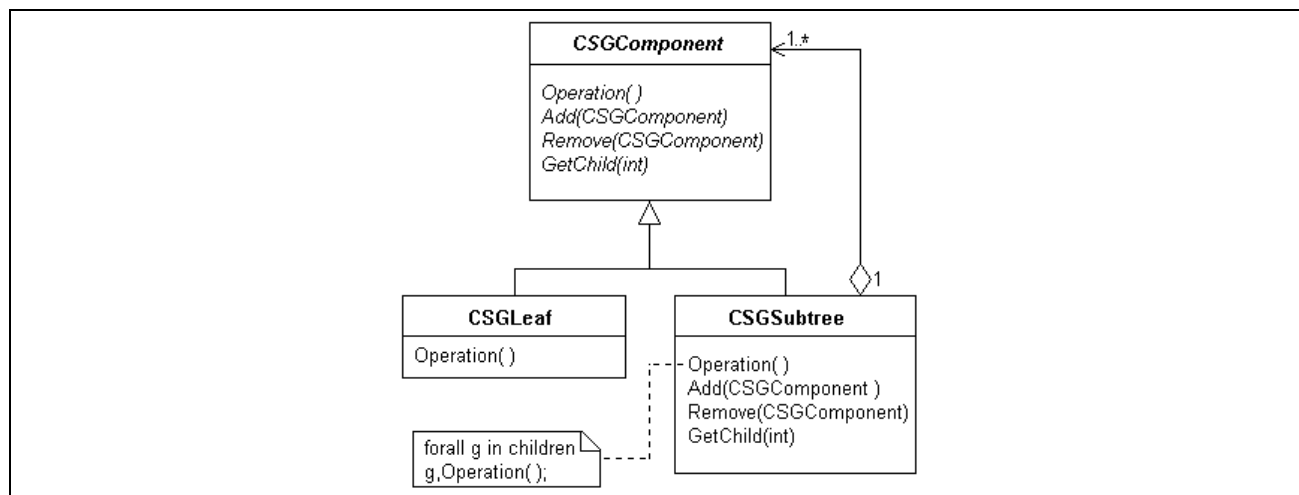
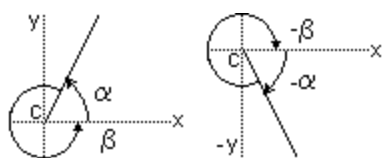
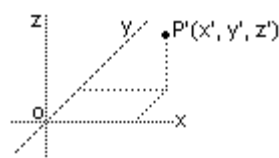
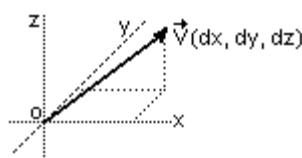
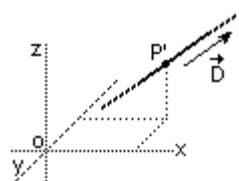
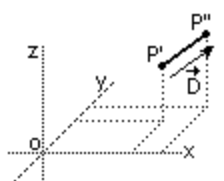
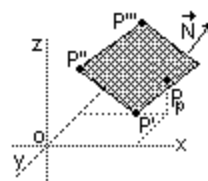



Figura IV.6 – O modelo de projeto para composição recursiva representando árvores CSG

IV.3 – PRINCIPAIS CARACTERÍSTICAS DE IMPLEMENTAÇÃO

O primeiro passo na implementação do Subsistema de Modelagem foi a geração dos elementos geométricos e das primitivas. As primitivas 2D permitiram definir perfis sobre os quais foram construídas as operações de varredura translacional e rotacional, gerando primitivas 3D. Paralelamente foram implementadas as transformações geométricas, a serem aplicadas tanto sobre perfis quanto sobre primitivas 3D. A estrutura básica da representação CSG foi então construída restando, por último, a implementação das operações booleanas e de montagem. Devido à complexidade do assunto e à escassez de tempo, optou-se por incluir no modelador a possibilidade de definir perfis compostos, que substituem alguns casos de operações booleanas, e não enveredar pelas particularidades de implementação dessas operações, deixando o sistema em condições de incorporá-las em uma próxima etapa.

As definições apresentadas para os elementos geométricos e as primitivas descritas foram baseadas em [Cen92, Dol85, Dol85a, Fer86, Rog90, Gom90]. A obtenção dos valores necessários para a construção das primitivas a partir dos parâmetros fornecidos foi possível com o uso dos conceitos de cálculo vetorial, geometria analítica planar e sólida e de trigonometria, encontrados em [Jud71, Ric72, Ayr76, Kin76, Lei77, Car78, Leh79, Iez85, Iez85a, Iez85b, Bou87, San98].

IV.3.1 – OS ELEMENTOS GEOMÉTRICOS BÁSICOS

Visando à criação e manipulação de primitivas, bem como à realização das operações de modelagem, visualização e seleção, foram implementados alguns elementos geométricos básicos, tais como: ângulos (*Angle*), planos (*Plane*), planos limitados (*RectPlane*), retas (*StraightLine*), segmentos de retas (*Segment*), vetores (*Vector3D*) e pontos (*Point3D*), detalhados no quadro IV.6. Esses elementos foram todos definidos no espaço tridimensional e implementados em módulos específicos para cada classe.

<table>
<tr><th colspan="2">ÂNGULO</th><th colspan="2">PONTO</th><th colspan="2">VETOR</th></tr>
<tr>
<td colspan="2">

- representa ângulo em graus, <=360°
- é definido pelo valor do ângulo mas pode ser construído por 2 pontos
- apesar de definido em 2D, pode ser aplicado no espaço tridimensional
- possui métodos para converter graus para radianos, calcular seno, coseno, tangente, realizar operações diversas (soma, divisão, comparações, etc.)</td>
<td colspan="2">

- representa um ponto no espaço
- é definido pelas coordenadas *x*, *y* e *z*
- pontos bidimensionais também são aqui definidos, porém com coordenada *z* nula
- possui métodos para converter seu valor em coordenadas cilíndricas e esféricas, calcular distância, obter ponto médio e circuncentro, verificar colinearidade, realizar operações diversas</td>
<td colspan="2">

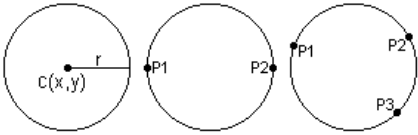
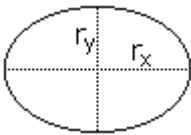
- representa um vetor com módulo, direção e sentido
- é definido por um vetor de 3 elementos contendo deslocamentos em *x*, *y* e *z* não simultaneamente nulos
- possui métodos para cálculo de norma, vetor unitário, produto vetorial e escalar; soma, subtração, divisão e multiplicação por escalar</td>
</tr>
<tr><th colspan="3">RETA</th><th colspan="3">SEGMENTO DE RETA</th></tr>
<tr>
<td colspan="3">

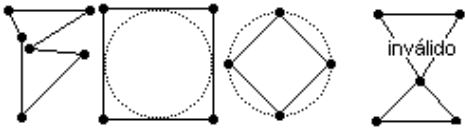
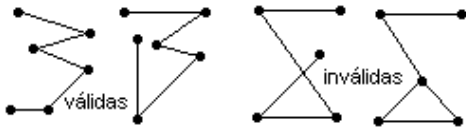
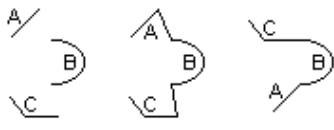
- representa uma reta infinita no espaço
- é definida por um ponto e um vetor direção não nulo
- possui métodos para verificar coplanaridade, concorrência e paralelismo com reta, calcular interseção com reta, ângulo entre duas retas, distância de um ponto à reta</td>
<td colspan="3">

- representa um segmento de reta
- é definida por seus pontos extremos, não coincidentes
- possui métodos para calcular interseção com outro segmento, verificar se contém ponto ou outro segmento
- como é filho de Reta, herda seus métodos</td>
</tr>
<tr><th colspan="3">PLANO</th><th colspan="3">PLANO LIMITADO RETANGULAR</th></tr>
<tr>
<td colspan="3">

- representa um plano de dimensões infinitas
- é definido pelo vetor normal não nulo e um ponto qualquer
- possui métodos para verificar paralelismo com reta e com plano, calcular interseção com reta e com plano, ângulo entre dois planos e distância de um ponto ao plano</td>
<td colspan="3">

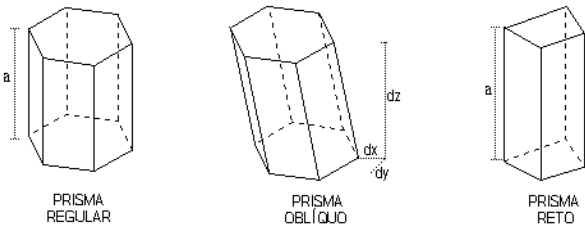
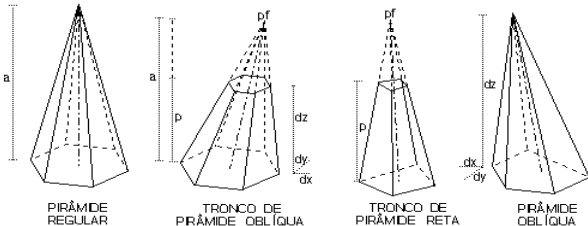
- representa uma superfície plana retangular limitada
- é definida por dois pontos opostos pela diagonal
- possui métodos para verificar se contém ponto, segmento de reta e interseções com outros planos limitados
- como é filho de Plano, herda seus métodos</td>
</tr>
</table>

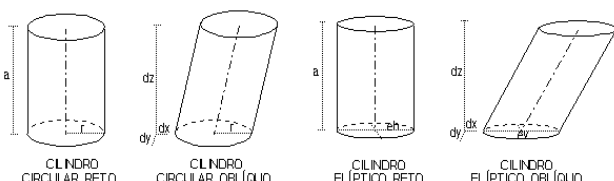
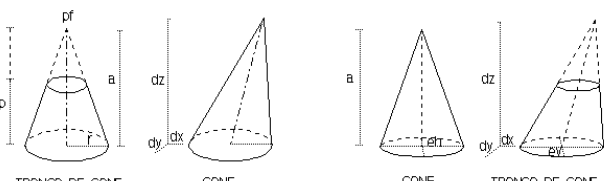
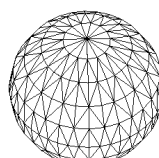
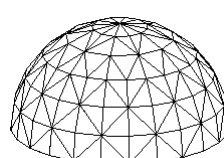
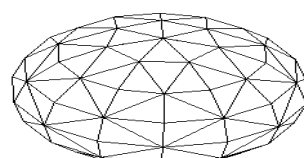
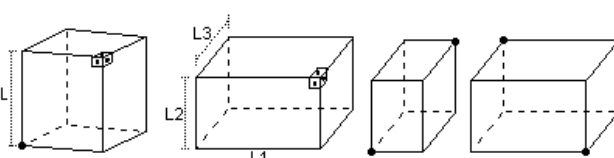
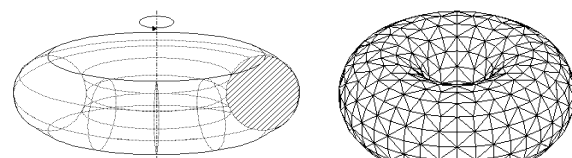
Quadro IV.6 – Elementos geométricos básicos definidos

IV.3.2 – PRIMITIVAS DISPONÍVEIS

O modelador oferece um conjunto finito de primitivas planares e sólidas, cujo tamanho, formato, posição e orientação são determinadas por um pequeno conjunto de parâmetros especificados pelo usuário. Visando obter maior facilidade descritiva, procurou-se ampliar as possibilidades de descrição, incorporando uma variada gama de parâmetros. As primitivas disponíveis estão definidas nos módulos *Primit*, *Primit2D* e *Primit3D* e serão detalhadas a seguir.

IV.3.2.1 – PRIMITIVAS PLANARES

As primitivas planares ou 2D são definidas sobre um plano *xy* fictício e posicionadas em relação à origem do sistema de coordenadas definido nesse plano. Em função do plano de trabalho utilizado e de seus parâmetros de definição, a primitiva é posicionada no espaço, recebendo as coordenadas definidas pelo usuário. As primitivas planares não fazem parte do modelo final obtido. Elas são definidas apenas para compor perfis a serem utilizados em operações de varredura, apresentadas na seção IV.3.3.

As primitivas planares disponíveis no modelador encontram-se relacionadas no quadro IV.7. Em termos de implementação, elas foram agrupadas da seguinte forma: círculos, elipses e arcos pertencem à classe *Arc*, que define arcos circulares ou elípticos de até 360º; retângulos, polígonos e polilinhas pertencem à classe *Polyline*, que é essencialmente uma seqüência de vértices aberta – vértices inicial e final desconectados – ou fechada – vértices inicial e final conectados; a classe *Silhouette* define um perfil sem auto-interseção, formado por uma lista de *Arc*s e *Polyline*s abertas e interligadas entre si, cujos vértices de ligação podem ser coincidentes ou não. Primitivas planares abertas podem também ser repetidas de forma radial, formando silhuetas mais complexas, para as quais se especifica o centro e o número de repetições.

CÍRCULO	ELIPSE
- representa os pontos eqüidistantes a um ponto (o centro) - pode ser definido pelo centro mais raio ou diâmetro, 2 pontos diametralmente opostos, 3 pontos quaisquer ou por duas tangentes e o raio	- representa os pontos eqüidistantes a dois pontos fixos, denominados focos - apesar da referência ser o foco, ela é mais facilmente obtida a partir dos raios horizontal e vertical

Quadro IV.7 – Primitivas planares disponíveis (início)

POLÍGONO	POLILINHA
- representa uma seqüência fechada de pontos coplanares, distintos, não colineares e sem auto-interseção - inclui o retângulo, definido por 2 vértices opostos - geralmente definido por uma seqüência de vértices - polígonos regulares inscritos ou circunscritos podem ser definidos pelo raio da circunferência mais o número de lados ou comprimento lateral	- representa uma seqüência aberta de pontos coplanares, distintos, não colineares e sem auto-interseção - é definido por uma seqüência de pelo menos dois vértices - não pode ser definida sozinha, apenas como componente de silhuetas - polígonos e polilinhas são definidas na classe *Polyline*, diferindo entre si por serem abertas ou fechadas
ARCO	SILHUETA
- representa um segmento de circunferência ou elipse - arco circular pode ser definido pelo ponto inicial mais: centro e ponto final, ângulo ou comprimento da corda; ponto final e ângulo, raio ou direção; outros dois pontos - arco elíptico é definido pelo centro, raios horizontal e vertical e ângulos inicial e final	- representa um perfil composto por arcos e segmentos coplanares e sem auto-interseção - é definido por uma lista de polilinhas e arcos, que serão interligados caso o ponto final de um componente não coincida com o ponto inicial do componente seguinte - dependendo da operação aplicada, ela poderá ser fechada

Quadro IV.7 – Primitivas planares disponíveis (continuação)

IV.3.2.2 – PRIMITIVAS SÓLIDAS

As primitivas sólidas disponíveis estão relacionadas no quadro IV.8. Elas são basicamente geradas a partir da varredura translacional ou rotacional de primitivas planares pré-definidas.

PRISMA	PIRÂMIDE
- sólido limitado por polígonos planos, com bases paralelas e congruentes e paralelogramos nas laterais - prisma regular: prisma com polígonos regulares nas duas bases - definido pelos parâmetros da base (polígono regular inscrito ou circunscrito, ou polígono qualquer) e pela especificação da translação reta ou oblíqua desta base - possui *n+2* faces, sendo *n* faces laterais e *2* bases, *n* arestas laterais, *3n* arestas e *2n* vértices	- sólido limitado por polígonos planos com uma das faces sendo um polígono e as demais triângulos com um vértice comum - pirâmide regular: pirâmide reta com polígono regular na base e triângulos isósceles congruentes nas laterais - tronco de pirâmide: gerado por uma pirâmide seccionada por um plano paralelo à base - possui *n+1* faces, sendo *n* laterais e *1* base, *n* arestas laterais, *2n* arestas e *n+1* vértices

Quadro IV.8 – Primitivas sólidas disponíveis (início)

CILINDRO	CONE
- sólido limitado por uma superfície cilíndrica fechada (obtida por uma reta geratriz paralela a um eixo percorrendo os pontos de uma linha diretriz) e dois planos paralelos cortando todas as geratrizes - é circular ou elíptico, conforme a diretriz seja uma circunferência ou uma elipse - um cilindro circular reto pode ser gerado pela rotação de um retângulo em torno de um eixo que contém um de seus lados. O mais freqüente, no entanto, é definir-se uma base circular (ou elíptica) e transladá-la em relação a um eixo - cilindros circulares geram prismas regulares inscritos ao serem aproximados por segmentos de reta.	- sólido limitado por uma superfície cônica fechada (obtida por uma reta geratriz que passa por um vértice e percorre uma linha diretriz) de uma só folha, e um plano que corta todas as geratrizes - é circular ou elíptico, em função de ser a diretriz uma circunferência ou uma elipse - tronco de cone: obtido ao seccionar um cone por um plano paralelo à base - possui com o cilindro a mesma relação que a pirâmide possui com o prisma - cones circulares geram pirâmides regulares inscritas ao serem aproximados por segmentos de reta.

ESFERA	HEMISFÉRIO	ELIPSÓIDE
- conjunto de pontos do espaço em que a distância de seu centro ao ponto do espaço é menor ou igual ao raio - é definida como o sólido gerado pela rotação de um semi-círculo em torno do eixo que contém o diâmetro	- conjunto de pontos do espaço pertencentes à metade superior ou inferior da esfera - é definido pelo centro, raio e lado, definido por um ponto localizado acima ou abaixo do centro	- conjunto de pontos contidos dentro da superfície definida ao girar uma elipse em torno de um de seus eixos - suas seções planas são todas elipses ou círculos - definida pelos seus raios em *x*, *y* e *z*

BLOCO	TORO
- paralelepípedo reto-retângulo contendo bases quadradas ou retangulares – constituem casos especiais de prisma - possuem ao todo *6* faces, *12* arestas e *8* vértices - cubo: possui todos os lados congruentes; é definido pelo comprimento lateral e coordenada inferior frontal esquerda - paralelepípedo: definido pelas comprimentos laterais ou pelos vértices opostos de sua diagonal principal	- sólido gerado pela rotação de um círculo ou elipse em torno de um eixo que lhe é externo e coplanar - se for utilizado um círculo, obtém-se um *toro de seção circular*, ou simplesmente toro; caso seja gerado a partir de uma elipse, denomina-se *toro de seção elíptica* - é definido pelo centro e raios horizontal interno e externo; se elíptico, será necessário fornecer também o raio vertical

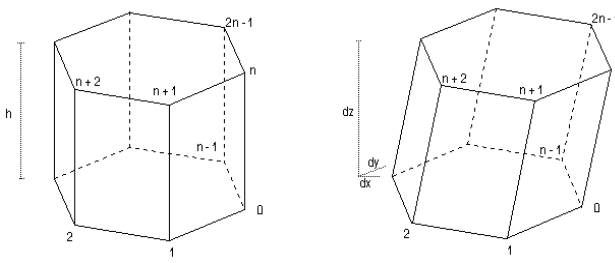
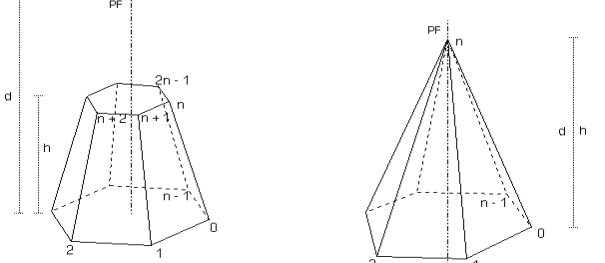
Quadro IV.8 – Primitivas sólidas disponíveis (continuação)

O quadro IV.9 apresenta os parâmetros para definição das primitivas apresentadas, bem como as verificações realizadas em cada caso. Os parâmetros para primitivas planares também foram incluídos neste quadro, uma vez que elas são utilizadas na definição das primitivas sólidas. A aproximação de sólidos curvos por facetas planares emprega o valor da tolerância atual utilizada, especificada na guia *Configurations* da caixa de diálogo aberta pela opção *Properties|Preferences*.

PRIMITIVA / PARÂMETRO	TIPO	DOMÍNIO	CONSISTÊNCIA
ARCO			
caso 1: coord. início, centro e fim	ponto	real	- início e centro não podem ser coincidentes; distância início-centro igual a centro-fim
caso 2: coord. início, centro	ponto	real	- início e centro não podem ser coincidentes
ângulo de varredura	ângulo	-2π a 2π	- ângulo diferente de zero
caso 3: coord. início, centro	ponto	real	- início e centro não podem ser coincidentes
comprimento da corda	valor	real +	- comprimento maior que 0 e menor que 2*raio
caso 4: coord início, outro ponto, fim	ponto	real	- pontos não podem ser colineares nem coincidentes
caso 5: dois elementos tangentes	elemento	no modelo	- elementos não podem ser coincidentes
raio	valor	real +	- valor maior que zero
POLILINHA			
coord. vários pontos	ponto	real	- mais de 2 pontos não colineares; os segmentos obtidos não podem possuir interseção
SILHUETA			
definida por arcos e polilinhas	diversos	diversos	- devem ser coplanares e não possuir interseção
CÍRCULO			
caso 1: coord. centro	ponto	real	- qualquer, dentro do plano de trabalho
raio	valor	real +	- valor maior que zero
caso 2: coord. centro	ponto	real	- qualquer, dentro do plano de trabalho
diâmetro	valor	real +	- valor maior que zero
caso 3: coord. 2 pontos	ponto	real	- pontos não coincidentes e diametralmente opostos
caso 4: coord. 3 pontos	ponto	real	- pontos não podem ser colineares nem coincidentes
caso 5: dois elementos tangentes	primit. 2D	no modelo	- elementos não podem ser coincidentes
raio	valor	real	- valor maior que zero
ELIPSE			
coord. centro	ponto	real	- qualquer, dentro do plano de trabalho
raios horizontal e vertical	valor	real +	- valores maiores que zero
POLÍGONO			
caso 1: coord. dois pontos	ponto	real	- qualquer, sendo extremos opostos na diagonal
caso 2: coord. vários pontos	ponto	real	- devem ser coplanares mas não colineares
caso 3: coord. centro circunferência	ponto	real	- qualquer, no plano de trabalho
raio inscrito/circunscrito	raio	real +	- valor maior que zero
número de vértice	valor	inteiro	- maior que 3
caso 4: coord. centro circunferência	ponto	real	- qualquer, no plano de trabalho
raio inscrito/circunscrito	valor	real	- valor maior que zero
comprimento lateral	valor	real	- menor que o lado de um triângulo equilátero inscrito/circunscrito à circunferência
BLOCO			
caso 1: comprimento lateral	valor	real +	- valor maior que zero
caso 2: comprimentos laterais	valor	real +	- valores maiores que zero
caso 3: coord. 2 pontos	ponto	real	- pontos não coincidentes e opostos pela diagonal
ESFERA			
coord. centro	ponto	real	- qualquer, no plano de trabalho
raio	valor	real +	- valor maior que zero
HEMISFÉRIO			
coord. centro	ponto	real	- qualquer, no plano de trabalho
raio	valor	real +	- valor maior que zero
lado	ponto	real	- coord. *x* do ponto diferente da coord. *x* do centro
ELIPSÓIDE			
coord. centro	ponto	real	- qualquer, no plano de trabalho
raios horizontal e vertical da elipse	valor	real +	- valores maiores que zero; o raio horizontal será utilizado para a varredura rotacional da elipse
PRISMA			
base: polígono qualquer	primit. 2D	disponível	- todas as definidas para gerar a primitiva
deslocamentos em *x*, *y* e *z*	vetor	real	- deslocamento normal à base não nulo

Quadro IV.9 – Parâmetros utilizados na definição de primitivas (início)

PIRÂMIDE base: polígono qualquer deslocamentos em *x*, *y* e *z* altura	 primit. 2D vetor valor	 disponível real real +	 - todas as definidas para gerar a primitiva - deslocamento normal à base não nulo - menor que o deslocamento normal à base, se nulo, indicará pirâmide completa
CILINDRO base: círculo ou elipse qualquer deslocamentos em *x*, *y* e *z*	 primit. 2D vetor	 disponível real	 - todas as definidas para gerar a primitiva - deslocamento normal à base não nulo
CONE base: círculo ou elipse qualquer deslocamentos em *x*, *y* e *z* altura	 primit. 2D vetor valor	 disponível real real +	 - todas as definidas para gerar a primitiva - deslocamento normal à base não nulo - menor que o deslocamento normal à base, se nulo, indicará cone completo
TORO raios horizontais interno e externo raio vertical (da elipse / círculo)	 valor valor	 real + real +	 - valores maiores que zero - valor maiores que zero

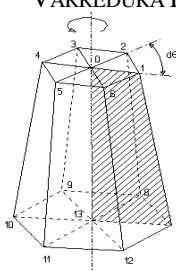
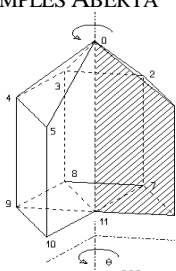
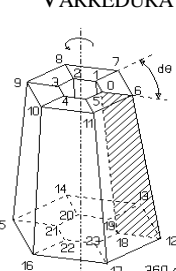
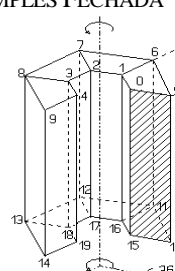
Quadro IV.9 – Parâmetros utilizados na definição de primitivas (continuação)

IV.3.3 – Operações de Varredura

Primitivas sólidas podem ser descritas a partir da varredura de primitivas planares, de forma translacional ou rotacional. A definição de primitivas por varredura permite aumentar em muito o potencial descritivo do modelador. As primitivas anteriormente definidas podem ser vistas como casos especiais desta abordagem mais ampla. As operações de varredura translacional e rotacional serão detalhadas a seguir, utilizando os conceitos obtidos em [Cas90, Fil87, Mor85, Req80]. Sua implementação compreende os módulos *Sweep*, *TraSweep* e *RotSweep*.

O princípio de varredura baseia-se na noção de mover uma região, denominada “gerador”, por uma trajetória, denominada “diretor”, utilizando duas formas básicas: a translacional – conhecida por *sweeping* – e a rotacional – conhecida por *swinging* –, que podem ser combinadas com o escalamento. Neste modelador, ambas utilizam uma face planar para definir o formato desejado. Na varredura translacional, a trajetória é linear e a face representa uma seção transversal, ao passo que na varredura rotacional, a trajetória é circular e a face corresponde a um perfil axial. Essas operações são essencialmente similares – o elemento básico é uma face percorrendo um caminho, enquanto faces laterais são criadas. A geometria da trajetória e das arestas que compõem a face a ser varrida determinam a geometria das faces laterais. As arestas laterais podem ser determinadas por meio do percurso varrido pelo vértice ou – mais caro e menos preciso – descobrindo a interseção entre as superfícies laterais geradas. Os vértices da face geradora devem estar orientados em um mesmo sentido, normalmente o sentido horário, para que seja possível definir corretamente o interior do sólido gerado e obter como resultado um sólido bem definido, limitado e com volume.

IV.3.3.1 – A VARREDURA TRANSLACIONAL

A varredura translacional ocorre ao longo de um eixo não coplanar ao plano de definição da face geradora. Valores positivos de deslocamento realizam a translação no sentido crescente do eixo diretor, e valores negativos realizam a translação em sentido decrescente. A varredura translacional de um perfil aberto ou com deslocamento coplanar à face geradora constituem casos especiais, que não geram sólidos. Existem basicamente dois tipos de varredura translacional: a simples e a cônica, explicadas no quadro IV.10. Baseado em [Cas90], o quadro IV.11 resume as principais características de cada tipo de varredura, aponta os parâmetros necessários e descreve a seqüência da numeração utilizada para as faces, considerando sua correta orientação para a definição do interior do sólido.

TRANSLACIONAL SIMPLES	TRANSLACIONAL CÔNICA
- a face geradora contém *n* vértices; pode ser reta ou oblíqua - translação reta: necessita da altura a ser atingida, uma vez que o deslocamento será perpendicular ao plano da face geradora - translação oblíqua: necessita dos deslocamentos em relação aos eixos coordenados, gerando um vetor não coplanar à face geradora - a face superior do sólido é obtida transladando a face geradora na direção do vetor ; estas faces são compostas por *n* arestas cada - os vértices são numerados de *0* a *2n-1*, fornecendo um total de *2n* vértices; para cada lado do polígono gerador surgirá uma face, resultando em *n* faces laterais e um total de *n+2* faces - as faces laterais serão sempre compostas por 4 arestas, formando retângulos para a translação reta e paralelogramos para a translação oblíqua, totalizando *3n* arestas	- os vértices da face geradora, além de serem transladados, convergem para um ponto (o ponto de fuga) - translacional cônica reta: o ponto de fuga está na posição do centróide do polígono gerador, a uma distância $d > 0$ - translacional cônica oblíqua: o ponto de fuga, não coplanar à face geradora, pode possuir qualquer deslocamento em relação aos eixos - se a altura *h* de translação for igual à distância *d* do ponto de fuga, a face superior será reduzida a um único vértice. Neste caso, as *n* faces laterais serão triangulares, o total de arestas será *2n* e o total de faces e vértices será *n+1*. Nos demais casos, sua configuração permanece igual à translacional simples. - além da face geradora, são necessários o deslocamento total em *x*, *y* e *z* e a altura de translação *h*

Quadro IV.10 – Tipos de varredura translacional

	TRANSLACIONAL SIMPLES	TRANSLACIONAL CÔNICA
PARÂMETROS	- face válida contendo *n* vértices - deslocamento em *x, y, z*	- face válida contendo *n* vértices - deslocamento em *x, y, z* e altura
FACES	*n* + 2	*n* + 1
ARESTAS	3*n*	2*n*
VÉRTICES	2*n*	*n* + 1
F - A + V	2	2
FACE INFERIOR	for (i=0; i<*n*; i++) {i};	for (i=0; i<*n*; i++) {i};
FACE LATERAL	for (i=0; i<*n*-1; i++) {*n*+i, *n*+i+1, i+1, i}; {2*n*-1, *n*, 0, *n*-1}	para translação completa for (i=0; i<*n*; i++) {i, *n*, (i+1) mod *n*}; para translação parcial, igual à translacional simples
FACE SUPERIOR	for (i=2*n*-1; i>=*n*; i--) {i};	para translação completa, não tem para translação parcial, igual à translacional simples

Quadro IV.11 – Principais características da varredura translacional

IV.3.3.2 – VARREDURA ROTACIONAL

A varredura rotacional ocorre em torno de um eixo coplanar e externo ao polígono gerador. Valores positivos para o ângulo realizam rotação no sentido anti-horário, enquanto valores negativos realizam rotação no sentido horário. Basicamente existem dois tipos de varredura rotacional: para polígonos abertos e para polígonos fechados, explicadas no quadro IV.12. Baseado em [Cas90], o quadro IV.13 resume as principais características de cada tipo de varredura, aponta os parâmetros necessários e descreve a seqüência da numeração utilizada para as faces, considerando o cuidado necessário com a orientação ao definir o interior do sólido.

VARREDURA ROTACIONAL SIMPLES ABERTA	VARREDURA ROTACIONAL SIMPLES FECHADA
- obtida pela rotação de um perfil aberto em torno do eixo formado pela reta que liga seu ponto inicial ao final; caso o perfil não possua pontos sobre o eixo, estes deverão ser incluídos - parâmetros: número de vértices do perfil gerador n, a resolução p (nº passos) e o ângulo total de rotação θ. - a cada passo, o número total de vértices é acrescido de n-2, ou seja, o número total de vértices da curva menos os dois pontos que estão sobre o eixo. Assim, o total de vértices é $(n-2)*p+2$. - de cada aresta da curva geradora resulta uma face no setor. Por ser aberta, o número de arestas da curva é n-1. Como as duas arestas limitantes da curva possuem um vértice sobre o eixo de rotação, na revolução as $2p$ faces geradas pelos extremos serão triangulares, e as $(n-3)*p$ restantes serão retangulares, totalizando $(n-1)*p$ faces e $(2n-3)*p$ arestas. - se o ângulo total de varredura for menor que 360°, aparecerão duas novas faces no início e no final da revolução. Nesse caso, o total de faces, arestas e vértices será, respectivamente, $(n-1)*p+2$, $(2n-3)*p+n$ e $(n-2)*p+n$.	- obtida pela rotação de uma face geradora em torno de um eixo que lhe é externo e coplanar - não deverá haver vértice da face geradora sobre o eixo de rotação, pois isso causaria multiplicidade do vértice, que fisicamente é único. - possui os mesmos parâmetros de definição que a aberta (ângulo de rotação θ, nº vértices n e nº passos p). - a cada passo ocorre um incremento angular $d\theta = \theta/p$, que define uma porção do sólido resultante, denominada *setor*. - como o polígono gerador é fechado, o número de arestas é igual ao de vértices (n) e cada aresta dará origem a uma face retangular, resultando em $n*p$ faces. São gerados n vértices a cada setor, totalizando $n*p$ vértices. O total de arestas é $2*n*p$. - quando o ângulo total de varredura for inferior a 360°, aparecerão duas novas faces, que devem ser adicionadas aos cálculos, assim como suas arestas e vértices. Neste caso, o número total de faces, arestas e vértices será, respectivamente, $n*p+2$, $(2*p+1)*n$ e $n*(p+1)$.

Quadro IV.12 – Tipos de varredura rotacional

	ROTACIONAL SIMPLES ABERTA	ROTACIONAL SIMPLES FECHADA
PARÂMETROS	- face válida com n vértices - rotação total (> 0) - resolução p (nº passos – definida pela tolerância)	- face válida com n vértices - rotação total (> 0) - resolução p (nº passos – definida pela tolerância) - eixo coplanar à face
FACES	parcial => $(n - 1)\ p + 2$ completa => $(n - 1)\ p$	parcial => $np + 2$ completa => np
ARESTAS	parcial => $(2n - 3)\ p + n$ completa => $(2n - 3)\ p$	parcial => $n\ (2p + 1)$ completa => $2np$

Quadro IV.13 – Principais características da varredura rotacional (início)

VÉRTICES	parcial => (n - 2) p + n completa => (n - 2) p + 2	parcial => n (p + 1) completa => np
F - A + V	parcial e completa => 2	parcial => 2, completa => 0
FACE INICIAL	parcial => {0}; for (i=(n-2)*p+n-1; i>0; i -=p+1) {i}; completa => não tem	parcial for (i=(n-1)*(p+1); i<=0; i-=p+1) {i}; completa => não tem

FACE LATERAL	parcial - triângulos superiores
	for (i=1;i<=p;i++)
	{0,i,i+1}
	parcial - faces retangulares
	for (i=1; i<(n-3)*(p+1); i+=p+1)
	{ for (j=0; j<p; j++)
	{ v1=i+j; v4=v1+1;
	v2=v1+p+1; v3=v2+1;
	{v1, v2, v3, v4} } }
	parcial - triângulos inferiores
	for (i=(p+1)*(n-3)+1; i<(n-2)*p+n-2; i++)
	{i, (n-2)*p+n-1, i+1};
	completa - triângulos superiores
	for (i=1;i<p;i++)
	{0,i,i+1};
	{0, p, 1};
	completa - faces retangulares
	for (i=1; i<(n-3)*p; i+=p)
	{ for (j=0; j<p-1; j++)
	{ v1=i+j: v4=v1+1;
	v2=v1+p; v3=v2+1;
	{v1, v2, v3, v4} } }
	{i+p-1, i+2*p-1, i+p, i}
	completa - triângulos inferiores
	for (i=p*(n-3)+1; i<(n-2)*p; i++)
	{i, (n-2)*p+1, i+1};
	{(n-2)*p, (n-2)*p+1, p*(n-3)+1}
		parcial
		for (i=0; i<n*(p+1); i+=p+1)
		{ for (j=0; j<p; j++)
		{ v1=i+j: v4=v1+1;
		v2=(v1+p+1) mod (n*(p+1));
		v3=v2+1;
		{v1, v2, v3, v4} } }
		
		completa => para todas as faces
		for (i=0; i<n*p; i+=p)
		{ for (j=0; j<p-1; j++)
		{ v1=i+j: v4=v1+p;
		v2=(v1+p) mod (n*p);
		v3=v2+1;
		{v1, v2, v3, v4} } }
		{i+p-1, (i+2p-1) mod (n*p), (i+p) mod (n*p), i} }

FACE FINAL	parcial {0}; for (i=p+1; i<(n-2)*p+n-1; i+=p+1) {i}; {(n-2)*p+n -1}; completa => não tem	parcial for (i=p; i<n*(p+1); i+=p+1) {i}; completa => não tem

Quadro IV.13 – Principais características da varredura rotacional (continuação)

IV.3.3.3 – CONSIDERAÇÕES SOBRE OPERAÇÕES DE VARREDURA

As varreduras translacional e rotacional são formas de descrição não ambíguas, mas não únicas, e o seu domínio está limitado a objetos com simetria translacional ou rotacional. O poder descritivo da varredura está intimamente relacionado à capacidade de descrição do perfil gerador. Por esse motivo, é interessante combinar polilinhas e arcos na construção do perfil, permitir a construção de perfis compostos por mais de um contorno, ou ainda, permitir a leitura de arquivo contendo a descrição de um perfil complexo. Tais características foram incorporadas ao modelador.

A matemática existente por trás da operação de varredura é mais complexa do que parece à primeira vista. Pouco se conhece sobre varredura aplicada a um caminho genérico, bem como sobre o movimento de um objeto que sofre deformações no decorrer da varredura. Podem ainda ocorrer problemas relacionados à degeneração, como a auto-interseção [Hof87], ilustrada na figura IV.7, ou a varredura por um caminho paralelo ao elemento gerador, que resultará em uma área, ao invés de um volume. Por esses motivos, optou-se por manter apenas a varredura por caminhos lineares e circulares.

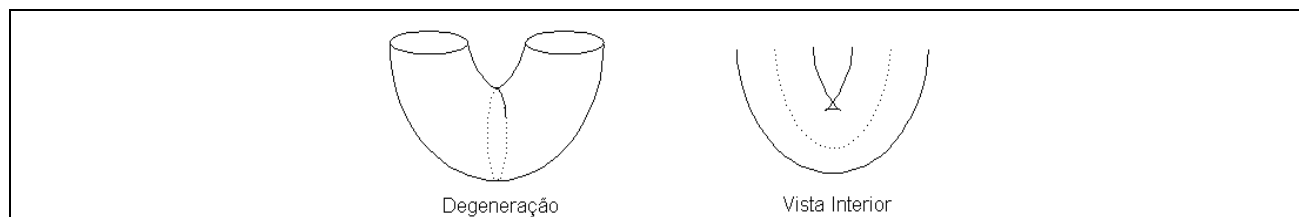



Figura IV.7 – Exemplo de degeneração em uma varredura genérica

IV.3.4 – AS TRANSFORMAÇÕES GEOMÉTRICAS

As transformações geométricas são intensamente utilizadas em modeladores de sólidos para definir, validar, calcular e posicionar objetos. As transformações geométricas básicas são a translação, a rotação e o escalamento. Transformações adicionais, como o espelhamento, podem ser obtidas pela composição destas transformações básicas. O módulo *GeoTransfOp* implementa as transformações geométricas bi e tridimensionais disponíveis no modelador, descritas no quadro IV.14 segundo [Fol90, Rog90, Hea86, Per90].

TRANSLAÇÃO	ROTAÇÃO
- movimentação de um objeto para uma outra posição no sistema de coordenadas, de modo que todos os pontos do objeto sejam deslocados de uma mesma distância em relação à sua posição anterior. - obtida adicionando-se as constantes de translação a todas as coordenadas que definem o objeto.	- movimentação de um objeto para outra posição, de modo que todos os pontos do objeto mantenham a distância em relação à origem que possuíam antes da transformação. - para a obtenção de uma rotação em torno de um ponto diferente da origem, é necessário fazer uma combinação de transformações de translação e de rotação, como ilustrado.

Quadro IV.14 – Transformações geométricas de interesse (início)

ESCALAMENTO	REFLEXÃO OU ESPELHAMENTO
- alteração no tamanho de um objeto, independentemente de ter ocorrido de maneira uniforme. - obtida pela multiplicação de todas as coordenadas que definem o objeto por fatores de escala não nulos. - um fator de escala maior que 1 provoca sua ampliação na direção do eixo afetado pelo fator; um fator de escala entre 0 e 1 provoca sua redução; um fator negativo provoca o espelhamento em relação ao eixo não afetado pelo fator. - quando fatores de escala iguais são aplicados em todas as direções, um objeto proporcional ao original é obtido; porém, quando fatores de escala diferentes são aplicados, obtém-se objetos com suas proporções alteradas. - é sempre realizado em relação à origem: fatores entre 0 e 1 o aproximam, enquanto fatores maiores que 1 o afastam. - para realizar o escalamento em relação a um ponto diferente da origem, deve ser feita uma combinação de transformações de translação e escala: translada-se o ponto de referência para a origem; realiza-se o escalamento e translada-se o ponto de referência para a sua posição original.	- transformação que produz a imagem espelhada (simétrica) do objeto gerada em relação a um eixo (plano) de reflexão. - em 2D, a reflexão é uma rotação de 180° que percorre o espaço e rebate a figura em relação a um eixo definido no plano inicial; para gerá-la em relação a um eixo coordenado, basta aplicar o fator de escalamento -1 ao outro eixo; para espelhar em relação à origem, utilizar -1 nos dois eixos. - a reflexão 3D equivale a uma rotação de 180° em relação a um eixo que percorre o espaço 4D e é rebatido no espaço 3D original; numa reflexão através do plano *xy*, só os valores da coordenada *z* serão alterados, recebendo o sinal inverso ao inicial; da mesma forma, nos planos *xz* e *yz*, só as coordenadas *y* e *x*, respectivamente, sofrerão inversão de sinal. - para refletir um objeto por um eixo (plano) diferente dos eixos (planos) coordenados, aplica-se translação e rotação até que o eixo (plano) coincida com um dos eixos (planos) coordenados; realiza-se a reflexão em relação a este eixo (plano); aplicam-se as transformações inversas à da etapa inicial, retornando o eixo (plano) à sua posição original.

Quadro IV.14 – Transformações geométricas de interesse (continuação)

Existem muitas vantagens em utilizar transformações geométricas. A mais importante delas é a possibilidade de aplicá-las a todos os pontos de um objeto aplicando-as apenas aos seus vértices. Outra grande vantagem é que transformações geométricas complexas podem ser realizadas a partir de outras mais simples. Para realizar, por exemplo, uma rotação no espaço em relação a um eixo de rotação qualquer – fornecidos o eixo e o ângulo de rotação – deve-se: transladar o objeto de forma que o eixo de rotação passe pela origem; rotacionar o objeto para que o eixo de rotação coincida com um dos eixos coordenados; realizar a rotação especificada; aplicar as rotações inversas para retornar o eixo de rotação a sua orientação original; aplicar a translação inversa para retornar o eixo de rotação para a sua posição original.

Ao invés de aplicar cada transformação ao objeto, é muito mais eficiente aplicar a composição dessas transformações de uma única vez. Surge aí um problema: a translação é tratada como uma soma, enquanto a rotação e a escala são tratadas como multiplicação. Para resolvê-lo, utilizam-se formas matriciais e coordenadas homogêneas [Rog90], que permitem tratar as transformações anteriores como multiplicações, combinando várias transformações antes de aplicá-las ao objeto e

realizando uma transformação complexa em uma única etapa. Com o uso de coordenadas homogêneas, as matrizes de transformação ficam definidas como apresentado no quadro IV.15. Segundo Rogers [Rog90], esta é a abordagem recomendada, pois geralmente é menos suscetível de erros e é computacionalmente mais eficiente que a abordagem matemática direta. Cuidado especial deve ser tomado com a seqüência de transformações a ser aplicada, pois em geral a multiplicação de matrizes não é comutativa, podendo levar a resultados diferentes em função da seqüência utilizada.

Obtém-se a inversa da matriz de translação invertendo o sinal das distâncias *dx*, *dy* e *dz*, o que produz a translação na direção oposta. Da mesma forma, obtém-se o escalamento inverso substituindo os fatores de escalamento *sx*, *sy* e *sz* por seus recíprocos *1/sx*, *1/sy* e *1/sz*. A matriz inversa gera o escalamento inverso, retornando à situação original. Já a matriz de rotação inversa é obtida substituindo o ângulo de rotação *a* por *-a*. Como somente a função *seno* é afetada pela mudança no sinal do ângulo, para obter a matriz inversa deve-se inverter somente o sinal dos coeficientes que envolvem o *seno* ou, de outra forma, calcular a matriz transposta da matriz de rotação original, pois $R^{-1} = R^{t}$.

	Translação	Escalamento	Rotação	Espelhamento
PLANO	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix}$	$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$	em x: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ em y: $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
ESPAÇO	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$	$\begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	eixo x: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ eixo y: $\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ eixo z: $\begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	plano xy: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ plano xz: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ plano yz: $\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Quadro IV.15 – Matrizes de transformação no plano e no espaço

IV.3.5 – Operações Booleanas

Apesar de não terem sido implementadas, as operações booleanas foram profundamente estudadas para a elaboração do projeto global do modelador, motivo pelo qual encontram-se detalhadas a seguir. As operações booleanas constituem a essência do modelamento CSG e estão fundamentadas na teoria dos conjuntos. O termo **conjunto** denota qualquer coleção de objetos bem definida. Objetos que pertencem a um conjunto são os seus elementos ou membros. Na modelagem

de sólidos, o elemento básico é o ponto. O **conjunto universo** contém todos os elementos de todos os conjuntos de interesse em uma dada situação, enquanto o **conjunto vazio** não possui nenhum elemento, sendo normalmente indicado por ϕ. Geralmente dois conjuntos A e B são ditos iguais quando possuem exatamente os mesmos elementos.

Novos conjuntos podem ser formados pela combinação de elementos de dois ou mais conjuntos. Dados os conjuntos A e B, pode-se construir um conjunto C cujos elementos são todos elementos de A, ou de B, ou de ambos, o que pode ser expresso por $C = A \cup B$. É importante observar que não existe repetição de elementos em C, mesmo que o elemento esteja contido tanto em A quanto em B, correspondendo ao operador lógico “ou inclusivo”. Se for construído um conjunto D contendo os elementos comuns entre A e B, o conjunto D será a interseção de A e B, expressa por $D = A \cup B$, significando o conjunto sem repetição dos elementos que se acham em ambos os conjuntos A e B. O complemento de um conjunto em relação ao conjunto universo E é o conjunto de todos os elementos de E que não são elementos de A, escrito como cA. Finalmente, se A e B são conjuntos, então $A – B$ denota o conjunto de elementos de A que não são elementos de B. Além disso, se o complemento é formado com relação ao conjunto universo E contendo os conjuntos A e B, então $A – B = A \cap cB$. As operações sobre conjuntos obedecem a certas regras, definindo propriedades pelas quais é possível combinar conjuntos, mostradas no quadro IV.16.

Propriedades	
Propriedades da União	
1. $A \cup B$ é um conjunto	propriedade do fechamento
2. $A \cup B = B \cup A$	propriedade comutativa
3. $(A \cup B) \cup C = A \cup (B \cup C)$	propriedade associativa
4. $A \cup \emptyset = A$	propriedade da identidade
5. $A \cup A = A$	propriedade da idempotência
6. $A \cup cA = E$	propriedade do complemento
Propriedades da Interseção	
1. $A \cap B$ é um conjunto	propriedade do fechamento
2. $A \cap B = B \cap A$	propriedade comutativa
3. $(A \cap B) \cap C = A \cap (B \cap C)$	propriedade associativa
4. $A \cap E = A$	propriedade da identidade
5. $A \cap A = A$	propriedade da idempotência
6. $A \cap cA = \emptyset$	propriedade do complemento
Propriedades do Complemento	
1. $cE = \emptyset$	O complemento do conjunto universo é o conjunto vazio
2. $c\emptyset = E$	O complemento do conjunto vazio é o conjunto universo
3. $c\,(cA) = A$	O complemento do complemento de um conjunto A é A
4. $c\,(A \cup B) = cA \cap cB$	Lei de De Morgan
5. $c\,(A \cap B) = cA \cup cB$	Lei de De Morgan
Propriedades Distributivas	
1. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	A união é distributiva em relação à interseção
2. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	A interseção é distributiva em relação à união

Quadro IV.16 – Propriedades da Teoria dos Conjuntos [Mor85]

Para os propósitos da modelagem geométrica, conjuntos consistem de pontos, e o conjunto universo E é o conjunto de pontos que definem o espaço Euclideano com a dimensão de interesse. A teoria dos conjuntos sugere métodos para operar sobre os pontos e classificá-los de acordo com propriedades do tipo "estar dentro", "fora" ou "na fronteira" de um sólido geométrico. Seja uma reta definida por um conjunto contínuo de pontos em E^1, no qual E^1 é o conjunto universo dos pontos considerados. Seja um subconjunto X de E^1 definido pelo segmento $a < X < b$, sendo a e b os pontos limítrofes do conjunto X, considerado aberto por não conter seus pontos limítrofes. Por outro lado, se X for definido pelo segmento $a \leq X \leq b$, então X será um conjunto fechado. O fechamento de um conjunto aberto é a união do conjunto aberto com o conjunto de todos os seus pontos limítrofes. Esses conceitos se estendem aos espaços e conjuntos de pontos em duas e três dimensões.

A fronteira de um conjunto fechado é o conjunto de todos os seus pontos limítrofes. De maneira oposta, o interior de um conjunto fechado é o conjunto de todos os pontos não pertencentes à sua fronteira. Assim, $X = bX \cup iX$, sendo bX o conjunto dos pontos pertencentes à fronteira de X e iX os pontos pertencentes ao seu interior.

Com esses conceitos em mente, é possível entender como formas simples são utilizadas na criação de formas mais complexas. Para esse fim, serão utilizados os operadores sobre conjuntos união ($\cup$), interseção ($\cap$) e diferença ($-$), denominados operadores booleanos, de acordo com as regras para a sua aplicação e combinação, processo este denominado álgebra booleana.

IV.3.5.1 – A NECESSIDADE DA REGULARIZAÇÃO

Uma característica que distingue os sólidos geométricos é o fato de serem definidos como conjuntos fechados, possuindo um subconjunto de pontos da fronteira e um subconjunto de pontos interiores. Operações booleanas similares à união, interseção e diferença de conjuntos são usadas para combinar objetos simples de modo a formar objetos mais complexos. Os algoritmos que realizam essas operações devem produzir como saída objetos que também são conjuntos fechados, possuindo subconjuntos disjuntos contendo pontos interiores e fronteira, além de preservar a dimensionalidade dos objetos iniciais. Este último requisito significa que em qualquer operação booleana, como $A \cup B = C$, todos os objetos devem ter a mesma dimensão espacial. Requicha e Voelcker [Req77, Req80] propuseram o uso de operações regularizadas sobre conjuntos, designadas por $\cup^*$, $\cap^*$ e $-^*$, que preservam a dimensionalidade e homogeneidade (objetos sólidos sem nenhuma parte solta ou pendente de dimensão menor). Tilove [Til80a] observa que a regularização significa tomar o que está dentro do conjunto e colocar uma capa fina envolvendo-o todo.

Conceitualmente, para computar $C = A \cap^* B$ dos objetos apresentados na figura IV.8 deve-se: computar $C = A \cap B$ padrão, cujo resultado é uma coleção de volumes, além de faces, arestas e vértices adicionais, que são estruturas de dimensão inferior e devem ser eliminadas; tomar o interior de $A \cap B$, que consiste de todos os pontos $p \in A \cap B$ de forma que uma esfera aberta de raio suficientemente pequeno ε centrada em p seja constituída somente de pontos de $A \cap B$; formar o fechamento desse interior pela adição de todos os pontos da fronteira adjacentes a algum ponto interior. É importante observar que estruturas de menor dimensão não cercam volumes, portanto não são adjacentes ao interior de $A \cap B$.

Na prática, porém, operações booleanas regularizadas não são implementadas dessa forma, mas sim classificando os elementos da superfície e eliminando estruturas de dimensão inferior. Essa classificação explícita só é realizada quando requerida ou quando ocorre a avaliação da fronteira, descrita no próximo capítulo.

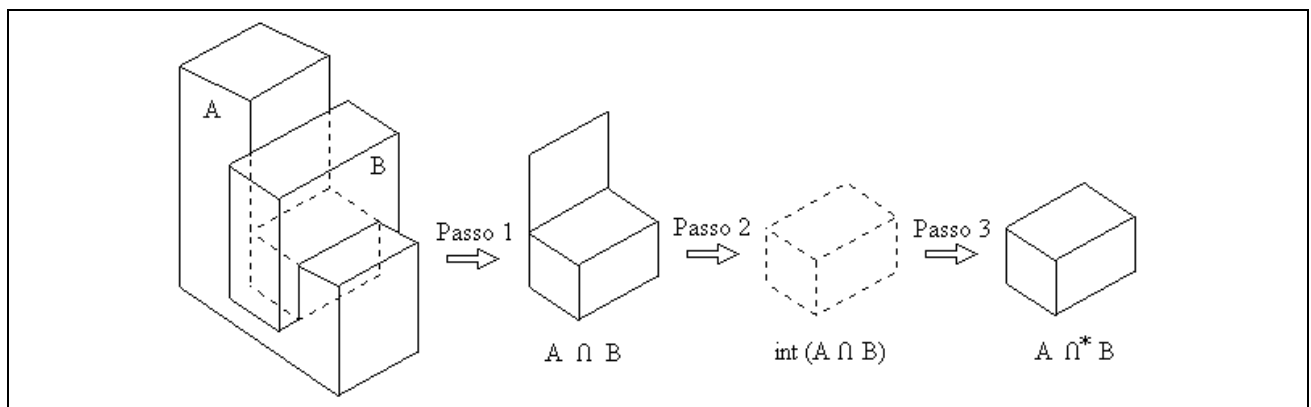



Figura IV.8 – Passos para a realização da interseção regularizada [Hof89]

É importante salientar que, ao executar várias operações booleanas em um conjunto de objetos, o resultado obtido dependerá da seqüência de operações aplicadas, e uma inversão na seqüência poderá originar resultado diferente. Vale ainda mencionar que os operadores booleanos são aplicados a objetos sólidos da mesma forma que a objetos bidimensionais. As operações booleanas regularizadas são portanto as mesmas, e tanto o fechamento quanto a homogeneidade dimensional são necessários para regularizar a operação. A figura IV.9 ilustra o efeito de várias combinações geradas pelas operações booleanas em dois objetos tridimensionais simples. Nas figuras IV.9a até c, nada resulta de não usual, mas nas figuras IV.9d a f, operações não regularizadas produzem interseções que não são tridimensionais. As operações regularizadas aplicadas corretamente nestes três últimos casos produzem resultados nulos. As operações booleanas regularizadas são detalhadas a seguir, seguindo a abordagem de Mortenson [Mor85].

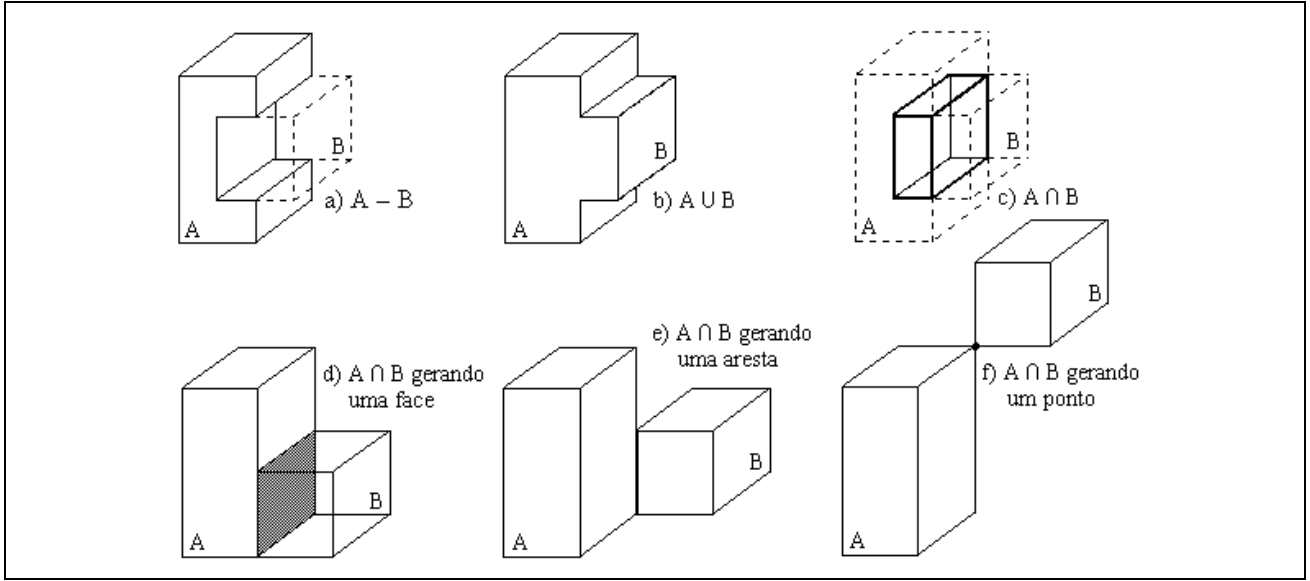



Figura IV.9 – Operações booleanas em sólidos tridimensionais [Mor85]

IV.3.5.2 – A Interseção Regularizada

Sejam dois objetos *A* e *B*, fechados e dimensionalmente homogêneos, expressos por:

$A = bA \cup iA$ $\qquad$ $B = bB \cup iB$

Para obter a interseção booleana regularizada, designada por A $\cap^*$ B, fechada e dimensionalmente homogênea, parte-se inicialmente da interseção padrão da teoria dos conjuntos:

$C = A \cap B$, que pode ser re-escrita como $C = (bA \cup iA) \cap (bB \cup iB)$, que se expande para

$C = (bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB)$

A interpretação geométrica de cada um dos quatro termos é ilustrada na figura IV.10, estando a direção de parametrização indicada pelas setas na figura IV.10a. Como $C = bC \cup iC$, é necessário encontrar os subconjuntos de *bC* e *iC* que formam um objeto fechado e dimensionalmente homogêneo, *C**. Os candidatos a *C** devem ser derivados dos termos da equação anterior. Na figura IV.10d, é possível identificar a parte interior de *C* e afirmar corretamente que:

$iC = iC^* = iA \cap iB$

A seguir, deve-se determinar *bC**, sendo $bC^* = Valid\ (bA \cup bB)$, com *Valid* indicando a regularização da expressão . Como a fronteira de qualquer novo objeto sempre será constituída de segmentos da fronteira dos objetos originais, pontos interiores não podem tornar-se pontos da fronteira, mas pontos da fronteira podem-se tornar pontos interiores. Para interseções regularizadas, tem-se $iA \cap bB \subset bC^*$ e $bA \cap iB \subset bC^*$.

Desse modo, já foram obtidas as porções correspondentes às figuras IV.10b a d da interseção. É necessário agora analisar *(bA ∩ bB)*, mostrado na figura IV.10a para determinar quais de seus subconjuntos são válidos na fronteira de *C**. O ponto marcado na figura IV.10a é um membro válido de *bC**, uma vez que ele é membro de *(iA ∩ bB)* e de *(bA ∩ iB)*. *(bA ∩ bB)* distingue-se dos demais casos por não possuir pontos interiores nem em A nem em B. Para verificar computacionalmente este fato, deve-se analisar a vizinhança do ponto de forma que ele possa ser corretamente classificado, seguindo os seguintes passos ilustrados na figura IV.10e:

1 - Em algum ponto P_1 no segmento *1*, criar dois novos pontos P_R e P_L perpendiculares a *1* em P_1 , situados a uma distância ε à direita e à esquerda, respectivamente, em relação à direção de parametrização. Fazer a mesma coisa em um ponto P_2 no segmento *2*.

2 - Para cada segmento, construir uma tabela verificando se os pontos P_R e P_L estão dentro de *A* e *B*. Para o segmento *1*, nenhum dos pontos está dentro de *A* e *B*, enquanto para o segmento *2*, o ponto P_L está dentro dos dois. Este teste determina que o segmento *2* é uma fronteira válida de *C**.

Seg. 1	in A	in B
P_R	0	1
P_L	1	0

Seg. 2	in A	in B
P_R	0	0
P_L	1	1

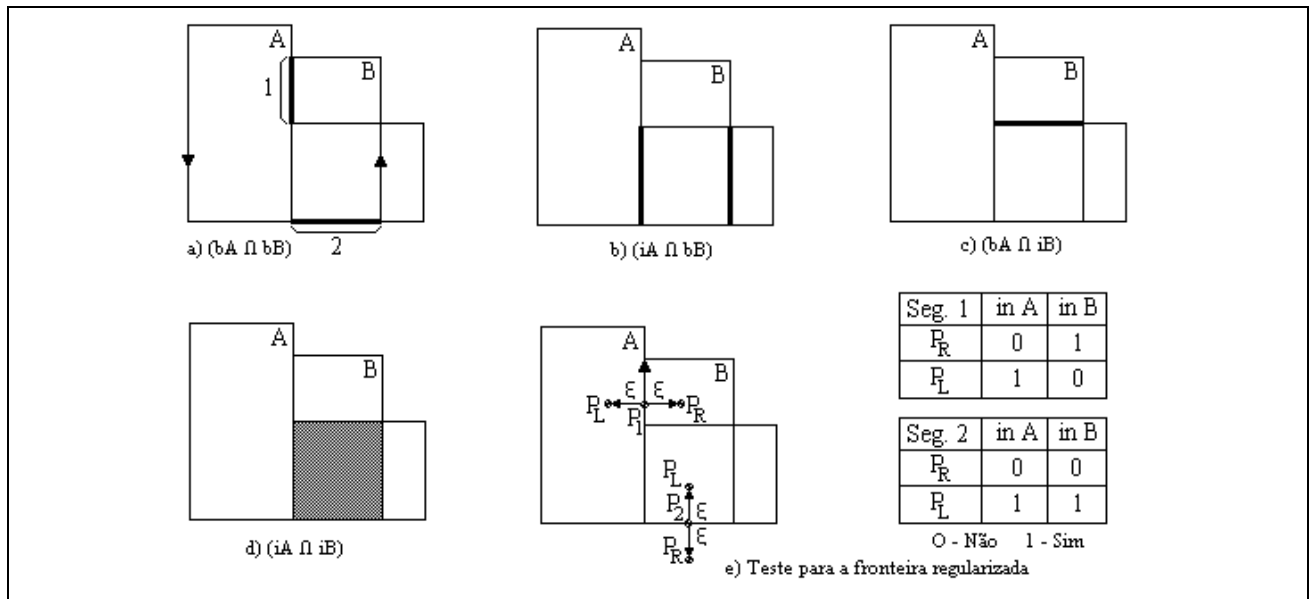



Figura IV.10 – Candidatos de uma interseção booleana regularizada e o teste da fronteira [Mor85]

Outra forma de classificação simples está disponível se for convencionada uma direção de parametrização. Em P_1, calcular o vetor tangente P_1^u da representação da fronteira de *A*, e P_1^v da representação da fronteira de *B*. Neste exemplo, eles possuem orientações opostas. Usando o mesmo processo para o segmento *2*, verifica-se que os vetores tangentes estão na mesma direção, de onde se conclui que se os vetores tangentes a um ponto em uma superfície sobreposta de dois objetos *A* e *B* possuírem a mesma direção, o segmento sobreposto estará na fronteira de $C^* = A \cap^* B$. Em caso contrário, o segmento não é uma fronteira válida.

Resumindo os resultados, tem-se a interseção regularizada de dois objetos A e B dada por:

$C^* = A \cap^* B$ sendo $C^* = bC^* \cup iC^* = Valid_b(bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB)$

Nada há nessa expressão que indique a dimensão, de onde se conclui que ela pode ser igualmente aplicada a objetos de uma, duas, três ou n dimensões.

IV.3.5.3 – A União Regularizada

Partindo dos mesmos objetos A e B, os componentes da união de A e B são mostrados na figura IV.11, constituindo o conjunto completo dos candidatos válidos para determinar $C^* = A \cup^* B$. Da mesma forma que com a interseção, expandindo a expressão da teoria dos conjuntos:

$C = A \cup B = (bA \cup iA) \cup (bB \cup iB) = bA \cup bB \cup iA \cup \underline{bB} \cup \underline{bA} \cup iB \cup \underline{iA} \cup \underline{iB}$

Os elementos redundantes estão sublinhados e podem ser desconsiderados. A equação fica:

$C = A \cup B = bA \cup bB \cup iA \cup iB$

Dentre esses componentes, é necessário determinar bC^* e iC^* para obter C^*. Nota-se que:

$iC^* = iA \cup iB \cup [Valid_i\ (bA \cap bB)]$

Como alguns pontos da fronteira tornam-se pontos interiores, é necessário incluí-los na expressão para que não ocorram buracos em iC^*. Cumpre notar-se que é redundante adicionar $\cup (bA \cap iB) \cup (bB \cap iA)$ ao lado direito da equação anterior. Observa-se a seguir que:

$bC^* = Valid_b\ (bA \cup bB)$, sendo:

$Valid\ bA =$ bA não presente em iB e parte em $bB = bA - [(bA \cap iB) \cup Valid_b\ (bA \cap bB)]$

$Valid\ bB =$ bB não presente em iA e parte em $bA = bB - [(bB \cap iA) \cup Valid_b\ (bA \cap bB)]$

Novamente, deve-se notar que existe uma ambigüidade em $(bA \cap bB)$ que deve ser resolvida por um teste similar ao discutido para o operador de interseção. Se todas as $(bA \cup bB)$ forem descartadas, bC^* estará incompleta. Assim, a fronteira do conjunto regularizado C^* é

$bC^* = bA \cup bB - [(bA \cap iB) \cup (bB \cap iA) \cup Valid_b\ (bA \cap bB)$

Conforme mostrado, $(bA \cap bB)$ é subdividido, sendo uma parte atribuída a iC^* e parte a bC^*. Além disso, nada é perdido, uma vez que $Valid_i\ (bA \cap bB) = Valid_b\ (bA \cap bB)$.

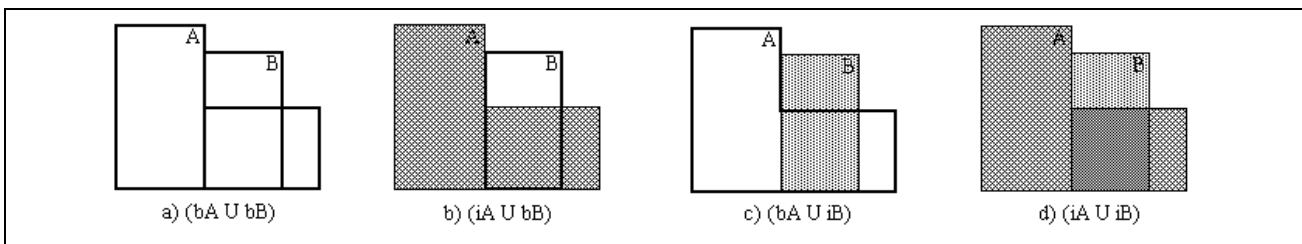



Figura IV.11 – Componentes candidatos de uma união booleana regularizada [Mor85]

IV.3.5.4 – A DIFERENÇA REGULARIZADA

Serão obtidos, finalmente, os componentes da operação de diferença regularizada $(A -^* B)$, mostrados na figura IV.12. Seguindo o mesmo raciocínio, expandindo a expressão sobre conjuntos:

$$C = A - B = (iA \cup bA) - (iB \cup bB) = (iA - iB) \cup (bA - iB) \cup (iA - bB) \cup (bA - bB)$$

$$C = (bA - bB - iB) \cup (iA - bB - iB)$$

Duas observações são claras a partir da figura IV.12. Primeiro, iC^* deve ser igual a $(iA - bB - iB)$ que, no caso do exemplo, resulta em dois conjuntos disjuntos. Em segundo, $C \neq C^*$, uma vez que certos segmentos de bC faltam em C. Se for adicionado $iA \cap bB$ a C, como na figura IV.12d, a fronteira estará ainda incompleta. O segmento faltante é um subconjunto de $bA \cap bB$. Novamente, um teste deve ser realizado para determinar o subconjunto válido. Para o caso da diferença, $Valid(bA \cap bB)$ são aqueles segmentos adjacentes somente a iC^* ou $(iA - iB)$. Assim,

$$bC^* = bC \cup (iA \cap bB) \cup Valid(bA \cap bB) = (bA - bB - iB) \cup (iA \cap bB) \cup Valid\ (bA \cap bB)$$

Logo, C^*, o resultado da operação regularizada para $(A - B)$, é

$$C^* = (bA - bB - iB) \cup (iA \cap bB) \cup Valid\ (bA \cap bB) \cup (iA - bB - iB)$$

Quando o objeto A envolve completamente B, a diferença $(A -^* B)$ resulta no modelamento de buracos.

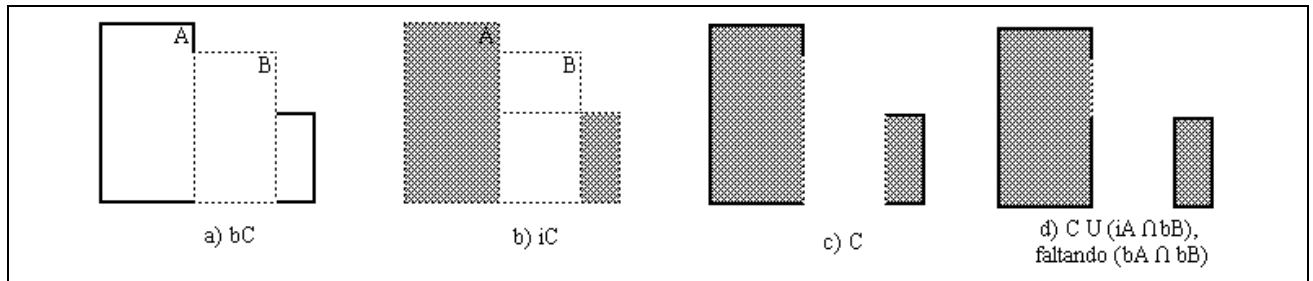



Figura IV.12 – Componentes candidatos de uma diferença booleana regularizada [Mor85]

IV.3.6 – A OPERAÇÃO DE MONTAGEM

Conforme já observado na seção IV.1.3, a operação de montagem é de grande interesse para este modelador, não só pelo encaixe e manipulação conjunta dos objetos que formam o modelo mas, principalmente, por permitir o estudo da interação de campos elétricos e magnéticos entre esses objetos, geralmente compostos por diferentes materiais. Para que esse estudo seja possível, é necessário interpretar de forma única cada fronteira existente entre os objetos componentes do modelo, assegurando, assim, a compatibilidade da malha de elementos finitos gerada.

IV.3.6.1 – O TRATAMENTO DE FRONTEIRAS INTERNAS

A fundamentação teórica utilizada em boa parte dos modeladores de sólidos é o conceito matemático de *r-sets*, como apresentado na seção I.3.1. Para um conjunto *X*, indica-se seu complemento, fechamento, fronteira e interior por *X'*, $\overline{X}$, *β(X) e int(X)*, respectivamente. Além de ser limitado e semianalítico, um *r-set X* é fechado e regular se for igual ao fechamento do seu interior, ou seja, $X = \overline{\text{int}(X)}$. Essa propriedade de fechamento, porém, cria dois problemas: primeiro, porque conjuntos fechados não podem modelar montagens de maneira conveniente, pois não é possível representar fronteiras compartilhadas por componentes de uma montagem como fronteiras internas de um conjunto fechado, e segundo, porque *r-sets* incluem sólidos cuja fronteira é de variedade múltipla. Considerando erros aritméticos, devido à precisão finita, isso pode levar a situações difíceis em aplicações nas quais a distinção entre montagem e peças isoladas é importante, como é o caso do eletromagnetismo.

O modelo de uma montagem deve refletir suas propriedades geométricas importantes, como forma, disposição e desconectividade de seus componentes. A maioria dos modeladores de sólidos baseados em CSG não suportam operações sobre montagens – que acabam sendo tratadas como se fossem componentes únicos – porém, ao realizar operações booleanas, as fronteiras compartilhadas entre componentes devem ser tratadas diferentemente das outras fronteiras.

Uma possível solução para o problema, descrita por Arbab [Arb90], trata as montagens como composição de *r-sets* e estende os operadores regularizados de forma a aceitar a operação de montagem, acoplando à estrutura CSG essas composições. É utilizada uma árvore CSG estendida, na qual os nós-folhas são *r-sets* e os nós-internos são operadores booleanos regularizados ou de montagem. Formalmente, porém, tais composições são apenas conjuntos de *r-sets* e não realmente *r-sets*, e as operações booleanas regularizadas não podem ser aplicadas sobre elas. Por meio de manipulações apropriadas [Arb90], porém, a árvore CSG estendida pode ser transformada, de forma que todas as operações de montagem estejam mais próximas da raiz e os operandos de todas as operações booleanas sejam *r-sets*, permitindo o uso dos mesmos operadores booleanos das árvores CSG padrão. Apesar disso, as montagens acabam permanecendo como expressões não avaliadas, sendo necessária sua reavaliação sempre que forem utilizadas, e superfícies compartilhadas entre dois *r-sets* participam desnecessariamente duas vezes de cada operação booleana, o que acaba sendo ineficiente para aplicações que envolvam montagens complexas. Mais sério do que a ineficiência, a redundância impede a interpretação de maneira única da fronteira, o que pode facilmente violar a integridade da montagem devido ao uso de precisão aritmética finita.

A principal razão para o problema é que uma propriedade topológica significativa – o compartilhamento de fronteiras comuns pelos *r-sets* componentes da montagem – não está diretamente disponível na representação, sendo expressa de forma indireta e aproximada. Além disso, os operadores regularizados do CSG não tomam conhecimento da situação especial de compartilhamento de fronteiras em montagens. Um formalismo de modelagem de sólidos para montagens deve representar suas fronteiras compartilhadas de forma confiável e fornecer operações booleanas que reconheçam e manipulem de forma apropriada essas fronteiras.

Arbab propõe um outro formalismo para o problema da montagem [Arb90], baseado em conjuntos regulares abertos – subconjuntos X de E^3 tal que $X = int(\overline{X})$, ou seja, que não incluem os pontos da fronteira mas admitem fronteiras internas. Conjuntos regulares abertos são duais de conjuntos regulares fechados, não possuindo segmentos de fronteira pendentes ou buracos infinitamente finos. A partir de *s half-spaces* – união de um número finito de conjuntos semi-analíticos regulares abertos –, Arbab define *s-sets* – *s half-spaces* limitados – e considera *s half-spaces* como superconjuntos próprios de conjuntos semi-analíticos regulares abertos. A generalização de conjuntos regulares abertos para *s half-spaces* permite separar fronteiras internas nos conjuntos. A fronteira interna de um conjunto *X*, designado $\iota(X)$, é o conjunto de pontos de sua fronteira que não são pontos da fronteira para o seu fechamento: $\iota(X) = \beta(X) - \beta(\overline{X})$. A figura IV.13 elucida este conceito.

As operações de modelagem utilizadas nesse formalismo são constituídas de operadores regularizados sobre conjuntos abertos, o operador de montagem e a interseção padrão. O operador de montagem é simplesmente uma forma restrita da união padrão, definida somente quando seus operandos são disjuntos.

a) um *s-set* válido e um *r-set* válido b) um *s-set* válido, mas não um *r-set* válido c) modelos inválidos

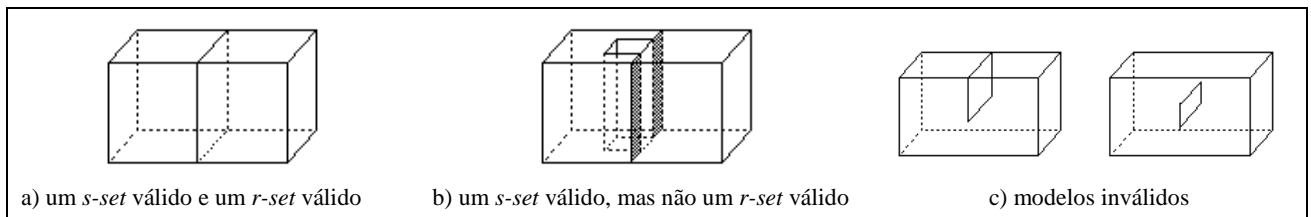



Figura IV.13 – Elucidação do conceito de *s-set*

As operações de modelagem para *s-sets* são expressas pelas seguintes equações [Arb90]:

$$R \cup^{+} S = int\left(\overline{R \cup S}\right) \qquad R + S = R \cup S \;\; if\, R \cap S = \varnothing \qquad R \cap^{+} S = int\left(\overline{R \cap S}\right)$$

$$c^{+}R = int\left(\overline{R'}\right) \qquad R -^{+} S = R \cap \left(c^{+}S\right)$$

Os *s half-spaces* são fechados para o conjunto de operações acima e para a interseção padrão. Se R e S são conjuntos regulares abertos, temos $R \cap^{+} S = R \cap S$. Interseções padrão e regularizadas (abertas) de *s half-spaces* com fronteiras internas, porém, não apresentam o mesmo resultado. Os *s-sets* são fechados para a interseção padrão e as operações acima, exceto para o complemento c^{+}, porque o *s half-space* resultante não possui fronteira. Conforme mostra a figura IV.14, a união padrão de dois *s half-spaces* não é necessariamente um *s half-space*. Como, porém, a operação de montagem realiza a união somente se os operadores são disjuntos, o resultado de uma montagem é sempre um *s half-space*. As operações regularizadas abertas de união ($\cup^{+}$), interseção ($\cap^{+}$) e complemento (c^{+}) sempre produzem conjuntos regulares abertos, destruindo assim todas as fronteiras internas. Somente o operador de montagem pode criar novas fronteiras internas. Os operadores de interseção padrão e diferença regularizada mantêm somente um subconjunto das fronteiras internas de seus operandos como fronteiras internas em seus resultados.

Ao invés de restringir o operador de montagem a uma função parcial (definida somente quando seus operandos são disjuntos, como na definição acima), pode-se definir o operador de montagem como uma função completa pela qual a diferença simétrica e a interseção são úteis:

$$R + S = (R -^{+} S) \cup (S -^{+} R) \cup (R \cap S)$$

Com esta nova definição, $R + S$ está sempre definida, e tanto *s-sets* quanto *s half-spaces* são fechados para a montagem. Obviamente, as duas definições para o operador de montagem produzem os mesmos resultados no caso em que $R \cap S = \emptyset$. A nova definição simplesmente divide a interseção de seus dois operandos em partes separadas e as adiciona à montagem resultante. Por exemplo, aplicada aos dois *s-sets* da figura IV.14a, a nova definição da montagem produz um *s-set* que parece ser a mesma figura, porém interpretada como possuindo todos os blocos componentes separados, de acordo com a figura IV.14c.

a) objetos iniciais b) União não resulta em um *s-set* c) Montagem resulta em um *s-set*

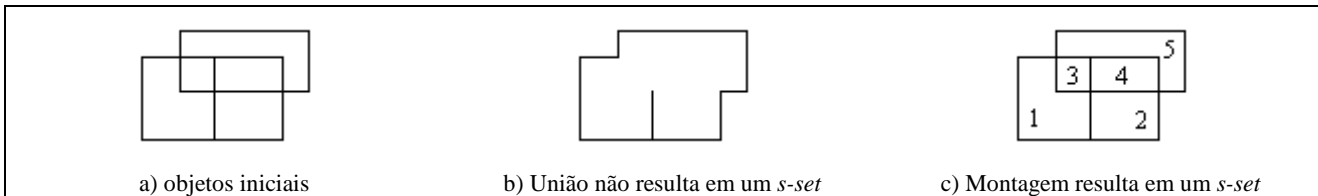



Figura IV.14 – A união e a montagem de dois *s-sets* (baseado em [Arb90])

A implementação das operações de modelagem descritas acima envolvem a interseção de superfícies da fronteira de seus operandos, a classificação das superfícies da fronteira resultantes e a combinação delas em um novo objeto. A função de classificação para *s-sets* é similar àquela para *r-sets*. Dado um *s half-space* H e uma superfície de fronteira S, a função de classificação $C(H,S)$

particiona S de maneira disjunta em três partes: a parte de S fora de H (*S out H*); a parte de S na fronteira de H (*S on H*); a parte de S dentro de H (*S in H*).

Formalmente, *C(S,H) = (S in H, S on H, S out H)* sendo:

$$S \text{ in } H = \overline{\text{int}^{S}\left(S \cap H\right)}^{s} \qquad S \text{ on } H = \overline{\text{int}^{S}\left(S \cap \beta(H)\right)}^{s} \qquad S \text{ out } H = \overline{\text{int}^{S}\left(S \cap c^{+} H\right)}^{s}$$

nessas expressões, S é um conjunto regular fechado e os superescritos $int^{S}(\overline{X})$ e $\overline{X}$ significam que o interior e o fechamento estão no universo de S. Seja $N_p(S)$ o vetor normal da superfície S no ponto p. A superfície inversa de S, designada $-S$, é a mesma superfície de S com normais opostas à superfície, ou seja, $N_p(-S) = - N_p(S)$ para todos os pontos p em S. Para a classificação *S on H* , as normais à superfície que são paralelas ou iguais entre si constituem casos de especial interesse. Para duas superfícies S e T, defini-se *S mates T* como a regularização fechada, no universo de S, do subconjunto das interseções cujas normais à superfície são paralelas - ou seja, possuem direções iguais ou opostas; *S shared T* é a regularização fechada, no universo de S, do subconjunto das interseções cujas normais à superfícies são iguais; a normal à superfície *S mates T* é definida como sendo a normal à superfície S. É importante notar que, enquanto *S shared T* e *T shared S* são sempre idênticas, *T mates S* e *S mates T* são ora idênticas, ora inversas entre si. Assim,

$$S \text{ mates } T = \overline{\text{int}^{S}\left(\left\{p \mid p \in (S \cap T) \wedge N_p(S) \parallel N_p(T)\right\}\right)}^{s}$$

$$S \text{ shared } T = \overline{\text{int}^{S}\left(\left\{p \mid p \in (S \cap T) \wedge N_p(S) = N_p(T)\right\}\right)}^{s}$$

A fronteira do *s half-space Z = X <op> Y*, sendo *<op>* uma operação booleana, pode ser derivada diretamente da fronteira dos dois *s half-spaces X* e *Y*. As definições no quadro IV.17 levam a uma implementação dos operadores sobre conjuntos que tratam montagens e peças conectadas de maneira semelhante e sempre "avaliam" seus resultados e retornam um *s half-space*:

$$\beta\left(X \cup^{+} Y\right) = \left\{\beta\left(\overline{X}\right) \text{ out } Y, \beta\left(\overline{Y}\right) \text{ out } X, \beta\left(\overline{X}\right) \text{ shared } \beta\left(\overline{\mathrm{Y}}\right)\right\}$$

$$\beta\left(X \cup Y\right) = \left\{\beta\left(X\right) \text{ out } Y, \beta\left(Y\right) \text{ out } X, \beta\left(X\right) \text{ mates } \beta\left(Y\right)\right\}$$

$$\beta\left(X \cap^{+} Y\right) = \left\{\beta\left(\overline{X}\right) \text{ in } Y, \beta\left(\overline{Y}\right) \text{ in } X, \beta\left(\overline{X}\right) \text{ shared } \beta\left(\overline{\mathrm{Y}}\right), \beta\left(\overline{X}\right) \text{ mates } \iota(\mathrm{Y}), \beta\left(\overline{Y}\right) \text{ mates } \iota\left(X\right)\right\}$$

$$\beta\left(X \cap Y\right) = \left\{\beta\left(X\right) \text{ in } Y, \beta\left(Y\right) \text{ in } X, \beta\left(\overline{X}\right) \text{ shared } \beta\left(\overline{Y}\right), \beta\left(\overline{X}\right) \text{ mates } \iota(\mathrm{Y}), \beta\left(\overline{Y}\right) \text{ mates } \iota\left(X\right), \iota\left(X\right) \text{ mates } \iota(\mathrm{Y})\right\}$$

$$\beta\left(c^{+} X\right) = \left\{-\beta\left(\overline{X}\right)\right\}$$

$$\beta\left(X -^{+} Y\right) = \left\{\beta\left(X\right) \text{ out } Y, -\left(\beta\left(\overline{Y}\right) \text{ in } X\right), \beta\left(\overline{X}\right) \text{ shared } \left(-\beta\left(\overline{\mathrm{Y}}\right)\right), \left(-\beta\left(\overline{Y}\right)\right) \text{ mates } \iota(\mathrm{X})\right\}$$

Quadro IV.17 – Definição dos operadores a serem aplicados sobre *s-sets* [Arb90]

A representação da fronteira de $\overline{X}$ é obtida da representação da fronteira de X removendo suas fronteiras internas. As definições de $\beta(X \cup^+ Y)$ e $\beta(X \cap^+ Y)$ são simétricas em X e Y. Assim, $X \cup^+ Y = Y \cup^+ X$ e $X \cap^+ Y = Y \cap^+ X$, como esperado. Como *S mates T* e *T mates S* podem ser o inverso um do outro, alguns segmentos de fronteira de $X \cup Y$ *e* $Y \cup X$ podem ter vetores normais opostos. Isto é possível somente para segmentos da fronteira que possuem os pontos interiores de X ou Y (ou ambos) de ambos os lados e assim são segmentos da fronteira interna no resultado. Se os sinais dos vetores normais das fronteiras internas não possuem significado na representação de montagens, $X \cup Y$ e $Y \cup X$ são os mesmos. Similarmente, a definição de $\beta(X \cap Y)$ é simétrica em X e Y, exceto para o termo $\iota(X)$ *mates* $\iota(Y)$. A única diferença possível entre $X \cap Y$ e $Y \cap X$ é, portanto, nos sinais dos vetores normais de algumas de suas fronteiras internas. Conseqüentemente, $X \cap Y$ e $Y \cap X$ representam o mesmo *s half-spaces*.

O complemento de um *s half-space* não pode ter fronteiras internas, o que simplifica a computação da diferença: ao invés de $R \cap (c^+ S)$, a expressão dada acima pode ser usada para descobrir a fronteira do resultado $R -^+ S$ diretamente. A figura IV.15 mostra o resultado de aplicar as operações acima em alguns conjuntos. A definição do operador de montagem usado nesta figura é o não restrito (que aceita interferência de operandos). As montagens podem ser usadas como padrões para a decomposição de objetos: a última linha na figura IV.15 mostra que um modelo para análise por elementos finitos poderia ser obtido pela interseção do objeto com um grid genérico usado como padrão de decomposição.

Considere-se agora a aplicação para *r-sets* dos procedimentos de avaliação da fronteira descritos acima. Se U e V são *r-sets*, é claro que $\beta(\overline{U}) = \beta(U)$, $\beta(\overline{V}) = \beta(V)$ e $\iota(U)$ e $\iota(V)$ são ambos vazios, e assim, $\beta(U \cup^+ V) = \beta(U \cup^* V)$. Para os *r-sets* U e V, $U \cup^* V = U \cup V$. As definições acima para $\beta(U \cup V)$ e $\beta(U \cup^+ V)$ não são idênticas, porém a definição para $\beta(U \cup V)$ pode produzir resultados sem sentido, já que o termo $\beta(U)$ *mates* $\beta(V)$ pode levar a segmentos de fronteira internos não existentes em *r-sets*. As definições para $\beta(U \cap^+ V)$ e $\beta(U \cap V)$ tornam-se idênticas e $\beta(U \cap^+ V) = \beta(U \cap V) = \beta(U \cap^* V)$. Da mesma forma, $\beta(c^+ U) = \beta(c^* U)$ e $\beta(U -^+ V) = \beta(U -^* V)$. Assim, o uso de *s-sets* ao invés de *r-sets* permite-nos representar montagens, bem como partes conectadas, e fornecer mais operadores para manipulá-los. Em contraste com $\cup^*$, $\cap^*$, c^* *e* $-^*$, as definições acima para $\cup^+$, $\cap$, $\cap^+$, c^+, $-^+$ e + levam a procedimentos que manipulam diretamente as fronteiras internas dos modelos *s-sets* para montagens.

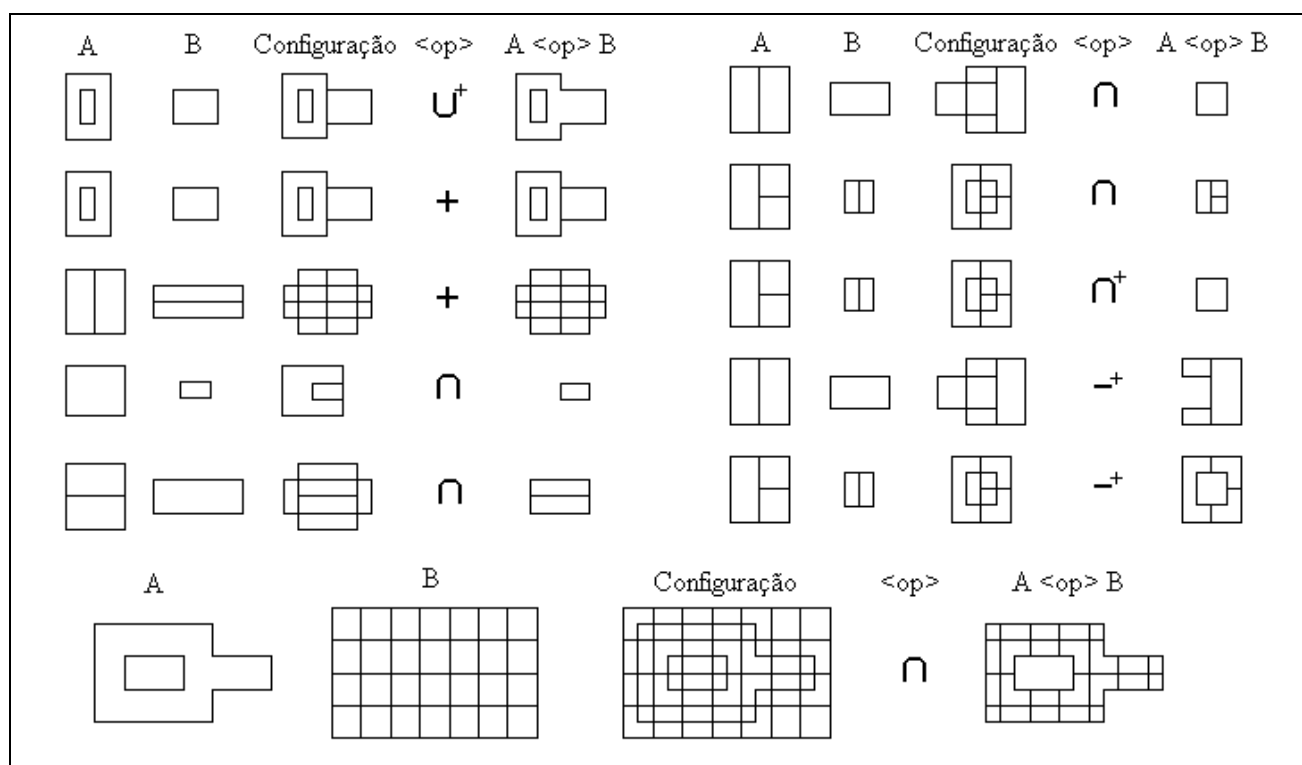



Figura IV.15 – Exemplos de operações booleanas e de montagem sobre *s-sets* [Arb90]

IV.3.7 – A Definição de Perfis Compostos

Após construídas, as primitivas 2D disponíveis no modelador podem ser combinadas entre si, formando perfis compostos por mais de um contorno, por meio da opção de menu *Create | 2D Primitive | Compound Silhouette*. Na definição de perfis compostos, é fundamental que exista um contorno externo – aberto ou fechado – envolvendo todos o demais, sem interceptá-los. De forma a garantir a geração de modelos válidos, os contornos internos são sempre fechados, e não podem envolver ou interceptar outros contornos do perfil, como mostra a figura IV.16. Além de aumentar o poder descritivo do modelador, a possibilidade de definir perfis compostos por mais de um contorno evita a realização de alguns casos de operações booleanas – as que retiram de um objeto partes contidas em seu interior, formando buracos.

a) perfis compostos válidos b) perfis compostos inválidos

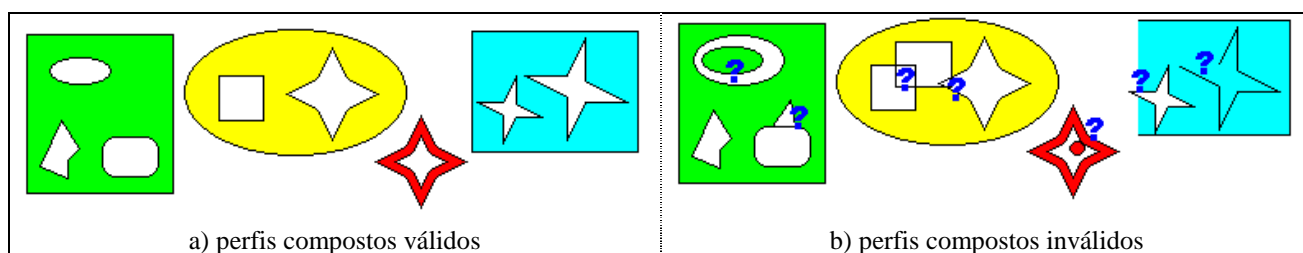



Figura IV.16 – Elucidação do conceito de perfil composto

IV.3.8 – A Estrutura CSG Utilizada

Definida nos módulos *CSG* e *CSGOp*, a estrutura CSG implementada neste modelador segue a estrutura CSG padrão detalhada no item IV.1.2, mas inclui a montagem, além das operações booleanas e transformações geométricas. Ela segue o padrão *Composite*, explicado no item IV.2.3, permitindo o uso de sub-árvores CSG em substituição a primitivas e evitando redundâncias na definição de sólidos.

Uma característica particular desse modelador é que todas as primitivas sofrem, logo após a especificação de seus parâmetros, um processo de triangularização, gerando uma representação de sua fronteira composta por facetas planares triangularizadas. Todas as operações de modelagem preservam esta característica. A presença exclusiva de faces triangulares facilita, de certa forma, o processo de avaliação da fronteira resultante de operações booleanas e de montagem aplicadas sobre componentes do modelo. Esse processo será detalhado no próximo capítulo.

"Por mais que se desenvolva a tecnologia,
faltará sempre ao computador o que só o criador tem: talento"
Paulo Malheiros

V – O SUBSISTEMA DE REPRESENTAÇÃO

A geometria sólida construtiva é considerada um esquema de representação não avaliado, por armazenar o processo de construção do sólido ao invés de sua descrição. Se por um lado esta característica proporciona técnicas de edição poderosas, como as descritas no capítulo anterior, por outro ela compromete a eficiência do modelo, uma vez que obriga sua reavaliação constante. É necessário, portanto, incorporar ao modelador uma representação avaliada, que mantenha o registro das informações sobre as fronteiras existentes no modelo e torne disponíveis imediatamente faces, arestas e vértices, elementos fundamentais para a geração da malha de elementos finitos.

Este capítulo é dedicado ao Subsistema de Representação, ou Núcleo, responsável principalmente pela geração e manipulação da estrutura de dados B-rep – *Boundary Representation* –, que descreve as fronteiras do modelo ora em construção. A seção V.1 apresenta as teorias e técnicas relacionadas à representação B-rep, fundamentais para compreensão deste subsistema. Após a especificação de requisitos, a seção V.2 explica a estrutura de dados B-rep desenvolvida para este modelador bem como as soluções orientadas para objetos adotadas. A seção V.3 detalha as principais características relacionadas à implementação da estrutura B-rep e demais estruturas envolvidas.

V.1 – O MODELAMENTO B-REP

A representação por fronteira, ou B-rep, baseia-se no princípio de que geometrias de maior dimensão são delimitadas por componentes de dimensões menores, criando uma hierarquia de componentes: um sólido é delimitado por uma superfície, que é representada pela união de faces, que são delimitadas por arestas, que por sua vez são delimitadas por vértices. Esta hierarquia define a topologia do objeto, representando um objeto deformável ou de borracha, que só fica rígido quando as informações geométricas – como equações das superfícies que definem as faces, equações das curvas que definem as arestas e as coordenadas dos vértices – são associadas, fornecendo rigidez ao objeto. Diferentemente do modelamento de superfícies, a representação B-rep garante que as superfícies do modelo definam uma partição do espaço completa, fechada e orientada, capaz de informar com exatidão a localização espacial do sólido.

A estrutura de dados escolhida para um modelador de sólidos por fronteira determina o conjunto de objetos que ele será capaz de representar adequadamente. A estrutura típica é formada por uma rede hierárquica de entidades topológicas, com vértices no nível mais baixo, compondo arestas, faces e objetos completos. Vértices, arestas e faces possuem ponteiros adicionais para entidades geométricas, denominadas pontos, curvas e superfícies, respectivamente. A figura V.1 ilustra essa estrutura e apresenta a relação entre as entidades topológicas. Alguns modeladores incluem níveis adicionais na hierarquia para especificar ciclos, ou *loops* – seqüência fechada de arestas representando as fronteiras de uma face – e cascas, ou *shells* – conjunto de faces conectadas que formam um sólido ou uma cavidade.

a) componentes básicos b) estrutura típica [Col97] c) relação entre entidades topológicas [Bae79]

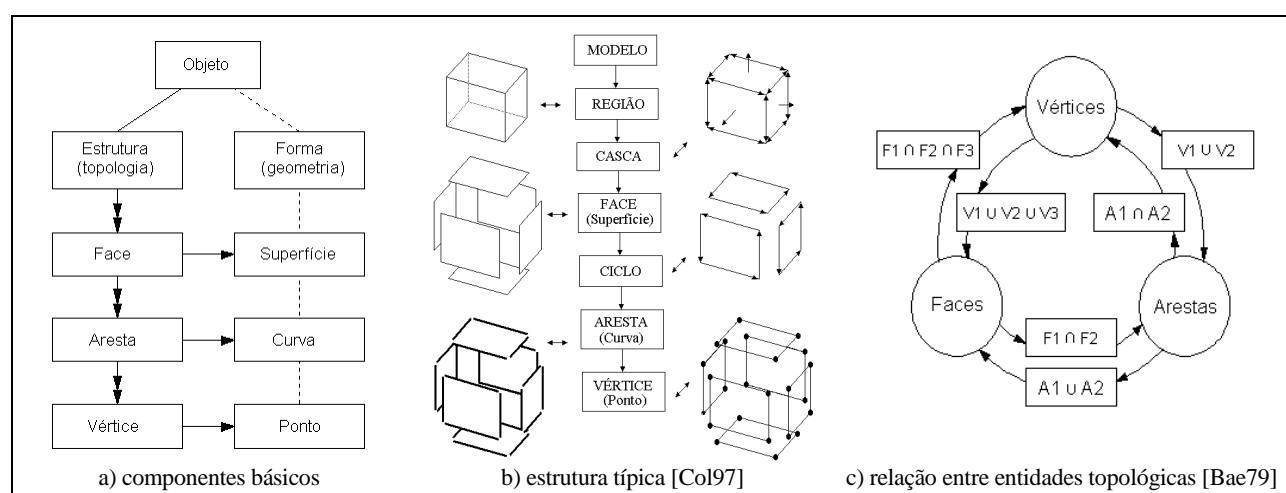



Figura V.1 – Características gerais da representação B-rep

V.1.1– PROPRIEDADES DA REPRESENTAÇÃO B-REP

Da mesma forma que o modelamento CSG, o esquema B-rep não garante a unicidade da representação, sendo possível a decomposição e representação de um mesmo objeto por diversas maneiras válidas e distintas. A representação B-rep é não ambígua, ou completa: desde que as faces sejam representadas de forma não ambígua, cada representação admitida pelo esquema corresponde a um único objeto, sempre interpretado da mesma maneira.

Sua potência descritiva varia em função das superfícies que podem ser utilizadas, mas de todo modo representa uma classe de objetos mais ampla que a modelagem construtiva. É, porém, bem menos concisa, pois armazena informações topológicas de forma explícita. Algumas implementações armazenam dados redundantes para acelerar algoritmos específicos, como relações de adjacência, enquanto outras utilizam métodos para tornar o modelo mais conciso. Se forem admitidas superfícies

curvas, a representação será mais precisa, mas o processo de validação poderá ser caro e complexo. Por outro lado, se superfícies curvas forem aproximadas por facetas planares, a representação perderá em precisão e concisão, mas a eficiência dos algoritmos envolvidos poderá ser melhorada.

A representação B-rep possui métodos poderosos, não implementáveis na modelagem construtiva, como as operações de modificação local – arredondamento e chanframento [Chi85, Jar85]. Em contrapartida, a complexidade dos algoritmos é elevada, variando em função da estrutura de dados implementada e dos critérios de validação automatizados. Sua validade é difícil de ser estabelecida, sendo necessário impor uma série de restrições geométricas e topológicas para garantir a integridade do modelo. É menos robusta que a abordagem construtiva, uma vez que a própria estrutura necessita de respostas a questões geométricas delicadas, relacionadas a coincidência ou paralelismo entre entidades geométricas, que podem levar a complicados problemas de tolerância.

A utilização de esquemas B-rep é muito ampla, devido principalmente à sua generalidade e ao fato de ser um modelo avaliado, que disponibiliza imediatamente os elementos necessários para manipulação e visualização. Representações B-rep são, porém, difíceis de serem construídas sem a ajuda do computador: a criação interativa de objetos válidos deixa a desejar, exigindo o fornecimento de facilidades adicionais para descrever objetos, visando garantir sua validade e consistência topológica. Modificações globais no modelo são difíceis de realizar, uma vez que o registro do processo de construção não é mantido.

Outras características importantes podem ser observadas na modelagem por fronteira: é adequada para gerar malhas de elementos finitos e saídas gráficas, por já dispor dos elementos necessários – vértices ou nós, arestas e faces; possui acesso eficiente às informações geométricas necessárias em muitas aplicações, como a geração automática de dados para a manufatura, e sua representação interna fornece meios para incluir no modelo dados não geométricos – como requisitos de tolerância, tratamento de calor, resistência, etc.

V.1.2 – Estruturas de Dados B-rep

Existem vários tipos de estruturas de dados para representar a geometria e a topologia de um modelo por fronteira, sendo as mais difundidas aquelas que se baseiam em arestas, ou seja, que utilizam a aresta como elemento de referência, representando faces em termos de ciclos de arestas e obtendo vértices indiretamente, a partir das arestas. A maioria das estruturas de dados baseadas em arestas são derivadas das estruturas Aresta-alada (*Winged-edge*) [Bau75] e Aresta-radial (*Radial-edge*)

[Wei87]. De uma forma geral, elas possuem complexidade variável, dependendo da quantidade de informações sobre relações de adjacência que armazenam, possuindo cada uma vantagens e desvantagens em termos de espaço para armazenamento, simplicidade dos algoritmos para manipulação e conjunto de sólidos admitidos. Em [Mag94] é apresentado um resumo das principais características, vantagens e desvantagens das estruturas mais conhecidas.

V.1.2.1 – ESTRUTURAS DE DADOS PARA SÓLIDOS DE VARIEDADE BIDIMENSIONAL

Entre os sólidos válidos, os que possuem maior interesse para a manufatura são os de variedade bidimensional (*manifolds*), apresentados na seção I.3.1. Tais sólidos possuem a característica de serem essencialmente bidimensionais, isto é, todo ponto pertencente à sua fronteira está rodeado por uma região "bidimensional" de pontos pertencentes à superfície, sendo possível modificar essa fronteira e torná-la um plano, sem que seja necessário rompê-la ou "colar" pontos separados uns dos outros. Tem-se, assim, um modelo planar – representação bidimensional da superfície de um objeto sólido, decomposta em faces –, que permite estudar os sólidos de variedade bidimensional e fornece uma abstração matemática útil e de aplicação direta na modelagem de sólidos por fronteira. Uma descrição matemática formal de modelos planares é apresentada por Mäntylä [Män88], e uma descrição formal dos conceitos de topologia relacionados pode ser encontrada em [Ale61]. A figura V.2 ilustra a topologia de um cubo maciço e de um cubo vazado – que são sólidos de variedade bidimensional – e os modelos planares correspondentes.

a) modelo planar de um cubo b) modelo planar de um cubo vazado

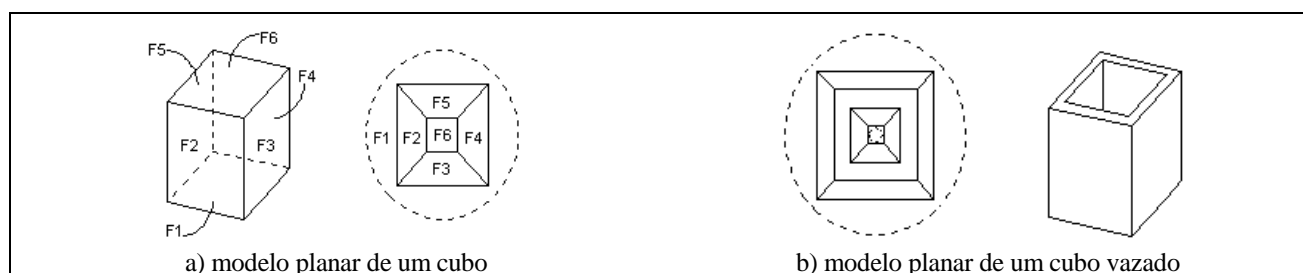



Figura V.2 – A topologia do cubo representada por modelo planar: cubo maciço e cubo vazado

Ao gerar sólidos válidos, as faces do objeto formam pelo menos uma casca fechada, separando seu interior de seu exterior. Para representá-los, Mäntylä propôs a estrutura *Half-edge* (Semi-aresta), derivada da Aresta-alada e utilizada no modelador GWB [Män82, Män88]. Essa estrutura foi inicialmente proposta para modelos poliedrais, mas pode ser estendida para admitir outros tipos de superfície, como quádricas, paramétricas, de forma livre, etc.

Na estrutura *Half-edge*, mostrada na figura V.3a, cada aresta do modelo é decomposta em duas semi-arestas, que são segmentos orientados de reta em um ciclo, correspondendo às ocorrências da aresta nos sentidos positivo e negativo das faces adjacentes, como apresentado na figura V.3b. Essa estrutura possui cinco níveis hierárquicos (Sólido, Face, Ciclo, Semi-Aresta e Vértice). Uma de suas vantagens em relação à Aresta-alada convencional é a de permitir que uma face possua várias bordas, devido à possibilidade de associar a cada face uma lista de ciclos. Quando uma face possui mais de uma borda, uma é considerada externa e as outras são consideradas anéis ou cavidades. Como resultado, um maior número de objetos pode ser modelado.

Uma variação da estrutura Semi-aresta proposta por Mäntylä foi desenvolvida para o modelador $(SM)^2$ [Mag94, Mag94b]. A principal diferença está na inclusão do elemento casca, que aumenta o poder descritivo do modelador por permitir a construção de sólidos com buracos e facilitar a execução de operações mais complexas, como as booleanas [Tsu92]. A figura V.3c apresenta uma visão hierárquica da estrutura de dados do $(SM)^2$. Atenção especial deverá ser dada à semi-aresta, elemento chave na construção correta de sólidos B-rep utilizando os operadores de Euler.

a) visão hierárquica [Män88] b) identificação de semi-arestas [Män88] c) variação da estrutura, utilizada no modelador $(SM)^2$

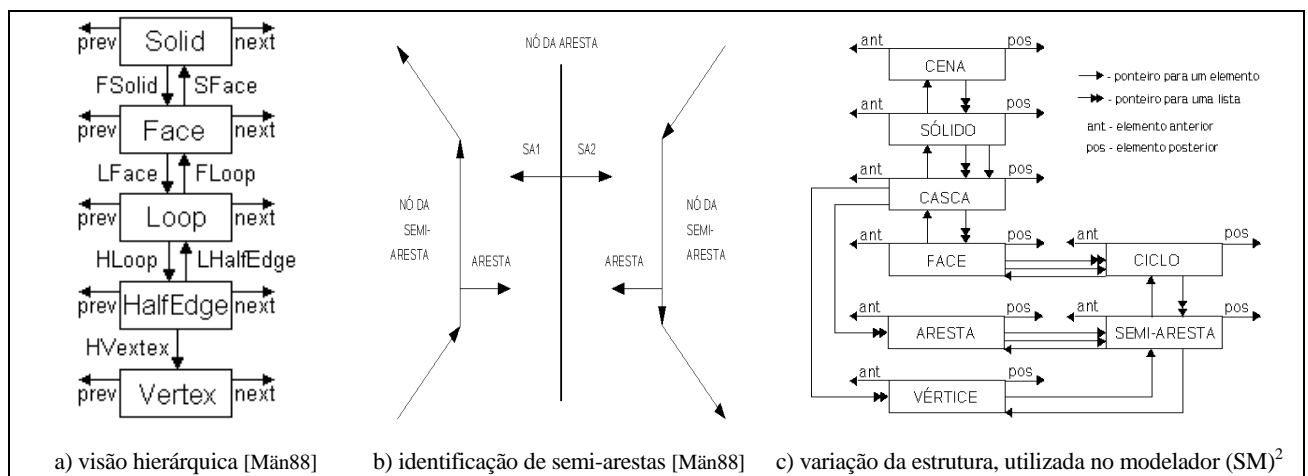



Figura V.3 – A estrutura de dados Semi-aresta (*Half-Edge*)

V.1.2.2 – ESTRUTURAS DE DADOS PARA SÓLIDOS DE VARIEDADE MÚLTIPLA

Um dos pontos chaves da abordagem B-rep para sólidos de variedade múltipla está na representação Aresta-radial (*Radial-edge*), proposta por Weiler, em oposição à Aresta-alada, estrutura clássica para sólidos de variedade bidimensional. Na representação Aresta-alada, uma aresta representa a fronteira exatamente entre duas faces, permitindo apenas uma conexão topológica entre um par de faces. Sua desvantagem é que um número infinito de planos ou faces podem interceptar-se em uma

única linha no espaço tridimensional. De uma forma mais abrangente, a representação Aresta-radial permite ligar topologicamente todas as faces que compartilham uma aresta em uma interseção linear. Weiler [Wei87] apresenta em detalhes essa estrutura de dados e operações relacionadas.

Os elementos topológicos básicos são vértice, aresta, ciclo, face, casca, região e modelo. A relação entre esses elementos é apresentada na figura V.4. Pode-se observar que, para qualquer elemento da hierarquia, existe um caminho direto daquele elemento para o elemento um nível acima e também para o elemento um nível abaixo.

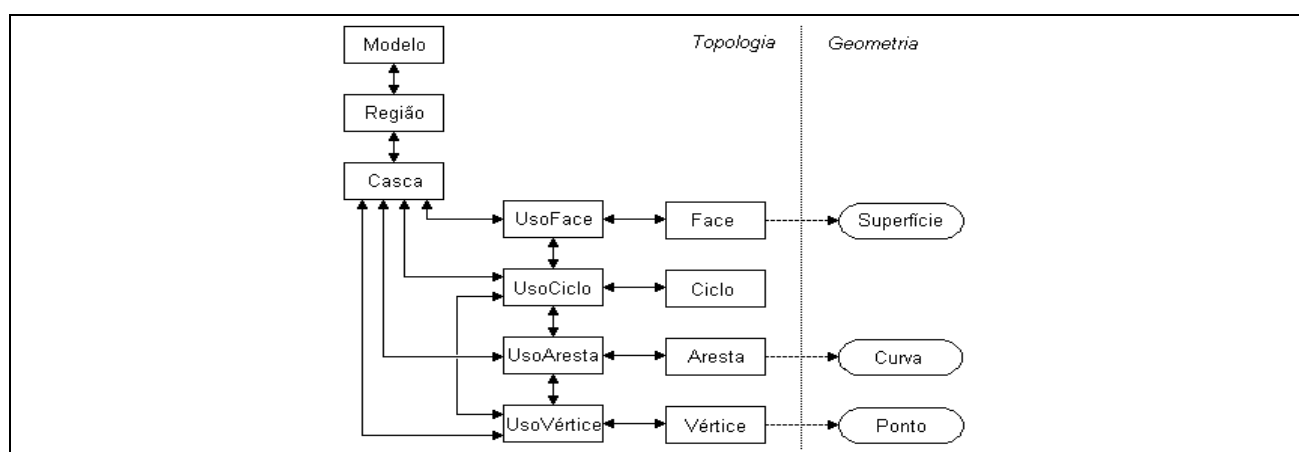



Figura V.4 – Hierarquia da estrutura de dados para sólidos de variedade múltipla (baseado em [Muu91])

O vértice representa um ponto topológico único. A aresta é uma linha ou curva delimitada por um único vértice ou dois vértices distintos. O ciclo é um único vértice ou um circuito de uma ou mais arestas, definindo um circuito ou uma fronteira no espaço. A face consiste em um ou mais ciclos, representando a área real da superfície. Ciclos externos são usados em faces para definir sua área, enquanto ciclos internos excluem a área da face, definindo buracos. Uma casca é um vértice único ou uma coleção de faces, ciclos e arestas. A coleção de faces em uma casca pode conter um volume (criando objetos fechados) ou representar superfícies arbitrárias. A região é uma coleção de cascas, e o modelo é uma coleção de regiões.

Para os elementos vértice, aresta, ciclo e face, existe uma distinção entre a existência do elemento e as ocorrências de seu uso, permitindo que múltiplos elementos topológicos compartilhem a mesma forma e geometria. Assim, um UsoVértice é uma ocorrência ou uso de um vértice, um UsoAresta é uma instância direta de uma aresta, um UsoCiclo é uma ocorrência de um ciclo e um UsoFace é uma instância ou uso de uma face. Cada lado da face é representado unicamente por um UsoFace, ou seja, toda face interna do modelo é referenciada por exatamente dois UsoFaces e toda face que delimita o modelo é referenciada por apenas um UsoFace – o correspondente ao lado que faz parte do modelo.

Note-se que cada elemento topológico referencia um elemento geométrico separado. Como resultado da separação entre a geometria e a topologia, os tipos de geometria suportados no modelador podem desenvolver-se para formas cada vez mais ricas, continuando a desfrutar de um conjunto comum de elementos topológicos com uma interface estável. Como exemplo, um modelador baseado inicialmente em faces planares pode ser expandido para suportar faces curvas, mantendo a mesma interface para a topologia. Muuss e Butler descrevem em detalhe essa estrutura [Muu91].

Várias foram as propostas de estruturas de dados para a representação de sólidos de variedade múltipla surgidas após o trabalho de Weiler: estrutura B-rep baseada em vértices [Cho89], complexo geométrico seletivo [Ros90] e introdução de entidades acopladas para relações de adjacência [Yam91]. Uma proposta interessante é a apresentada por Lee [Lee93, Han96], consistindo em uma representação B-rep de variedade múltipla hierárquica compacta, obtida a partir da estrutura *Half-edge* pela introdução dos elementos topológicos parciais, que representam as condições de variedade múltipla em torno de um vértice, aresta ou face, como apresentado na figura V.5.

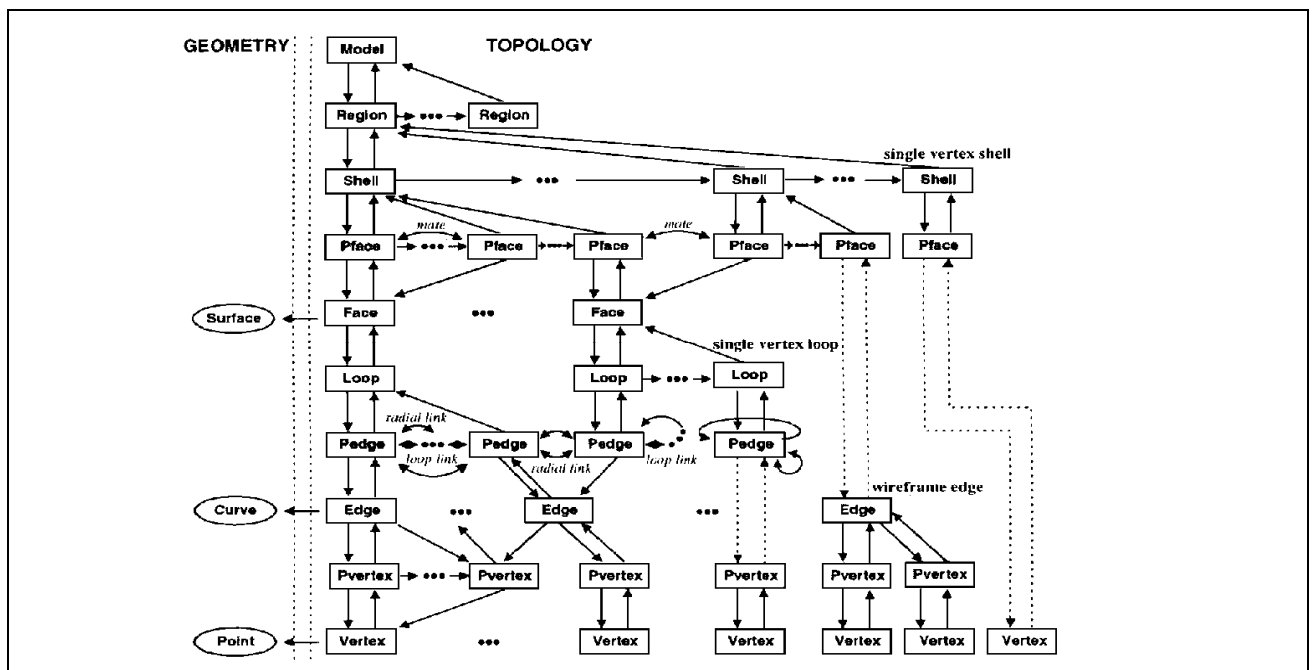



Figura V.5 – Estrutura de dados para sólidos B-rep de variedade múltipla proposta por Lee [Lee93]

Este esquema de representação possui os elementos topológicos primitivos *model*, *region*, *shell*, *face*, *loop*, *edge* e *vertex*. Três elementos topológicos parciais são introduzidos para representar as relações de adjacência entre os elementos topológicos primitivos: *Pface* define a fronteira das regiões contidas em cada lado da face; *Pedge* representa a relação de adjacência entre um ciclo e uma aresta e contém duas ordens cíclicas – a ordem das arestas no ciclo (*loop link*) e a ordem das faces em torno das arestas (*radial link*); *Pvertex* é uma entidade topológica única,

representando a relação topológica entre um vértice e as estruturas de variedade bidimensional a ele adjacentes – para uma condição de variedade múltipla em torno de um vértice, o número de vértices parciais a ele associados é igual ao número de variedades simples e arestas fio-de-arame adjacentes a ele. Os elementos topológicos parciais simplesmente representam as relações de adjacência entre os elementos topológicos primitivos, não contendo nenhuma informação própria sobre a fronteira, o que diferencia esta proposta das apresentadas por Weiler, Choi e Yamaguchi [Han96].

V.1.3 – Construção de Sólidos B-rep pelos Operadores de Euler

Um sólido B-rep pode ser considerado topologicamente correto se os seus elementos estão conectados de forma apropriada (ex.: faces são delimitadas por arestas que conectam dois vértices). A garantia da validade topológica é fornecida pelos Operadores de Euler, descritos a seguir.

V.1.3.1 – Operadores de Euler Aplicados a Sólidos de Variedade Bidimensional

Um dos principais resultados da teoria desenvolvida para modelos planares é o Teorema da Invariância [Ale61], que garante às propriedades topológicas manterem-se constantes para todos os modelos planares que podem ser construídos para sólidos de variedade bidimensional. A idéia persiste para modelos por fronteira, com o emprego dos "operadores de Euler", derivados da conhecida "Lei de Euler": em qualquer poliedro simples, a relação entre o número de faces (F), arestas (E) e vértices (V) deve satisfazer à equação:

$V – E + F = 2$

A fórmula pode ser generalizada para sólidos de variedade bidimensional arbitrários, pela introdução de três outros parâmetros: o número total de anéis – cavidades em faces – existentes no sólido (R, de *Ring*); o número total de orifícios que atravessam o sólido (H, de *Hole*); o número de componentes conexos – cascas – que compõem o sólido (S, de *Shell*").

$V – E + F = 2 (S – H) + R$

As operações de Euler trabalham sobre a topologia do modelo. Sempre que um operador de Euler é aplicado, o sólido resultante satisfaz a equação, sendo topologicamente válido. A geometria é incorporada à representação à medida que os vértices vão sendo criados e recebem suas coordenadas. Braid *et al.* [Bra80] mostraram que cinco operadores e seus inversos (que permitem desfazer a operação) são suficientes para descrever qualquer sólido de variedade bidimensional, satisfazendo a

equação anterior. De fato, estes cinco operadores podem ser escolhidos de diferentes formas, e alguns autores [Bau75, Eas79, Bra80, Män82] propuseram pequenas variações na coleção de operadores selecionados. Os operadores de Euler propostos por Mäntylä e Sulomen [Män82] são apresentados no quadro V.1.

Pesquisas desenvolvidas no início dos anos 80 por Mäntylä [Män82, Män84] mostram que qualquer representação por fronteira inserida no espaço Euclidiano pode ser construída a partir dos operadores de Euler, e que sua utilização satisfaz a todas as condições combinatórias para validade topológica de objetos de variedade bidimensional. Os operadores de Euler, portanto, fornecem um meio natural de manipular representações B-rep para sólidos de variedade bidimensional.

OPERADOR	SIGNIFICADO	V	E	F	H	R	S
MVFS	*Make Vertex, Face, Solid* Constrói vértice, face, sólido	+1	0	+1	0	0	+1
KVFS	*Kill Vertex, Face, Solid* Destrói vértice, face, sólido	-1	0	-1	0	0	-1
MEV	*Make Edge, Vertex* Constrói aresta, vértice	+1	+1	0	0	0	0
KEV	*Kill Edge Vertex* Destrói aresta, vértice	-1	-1	0	0	0	0
MEF	*Make Edge, Face* Constrói aresta, face	0	+1	+1	0	0	0
KEF	*Kill Edge Face* Destrói aresta, face	0	-1	-1	0	0	0
MEKR	*Make Edge, Kill Ring* Constrói aresta, destrói anel	0	+1	0	0	-1	0
KEMR	*Kill Edge, Make Ring* Destrói aresta, constrói anel	0	-1	0	0	+1	0
MFKRH	*Make Face, Kill Ring Hole* Constrói face, destrói anel e cavidade	0	0	+1	-1	-1	0
KFMRH	*Kill Face Make Ring Hole* Destrói face, constrói anel e cavidade	0	0	-1	+1	+1	0

Quadro V.1 – Operadores de Euler propostos por Mäntylä e Sulomen

V.1.3.2 – OPERADORES DE EULER APLICADOS A SÓLIDOS DE VARIEDADE MÚLTIPLA

A maioria dos sistemas de modelagem para sólidos de variedade múltipla também utilizam os operadores de Euler para manter a integridade topológica dos modelos, implementando sobre eles as operações de modelagem de alto nível. De várias pesquisas, como as de Yamaguchi [Yam91], Masuda [Mas93] e Lee [Lee93], derivaram-se fórmulas de Euler consistentes, que podem ser aplicadas a modelos de variedade múltipla, bem como operadores de Euler, aplicáveis às estruturas topológicas correspondentes. Os operadores de Euler de interesse para este trabalho são

os definidos por Lee, acompanhando a estrutura de dados por ele proposta. A fórmula básica que pode ser aplicada à estrutura de dados topológica é:

$$V - E + F - L = S - C + R$$

V, E e F são vértices, arestas e faces; L é o número de ciclos definindo buracos, S é o número de cascas vazias nas regiões, C é o número de ciclos que não podem ser contraídos até um ponto e R é o número de regiões. A partir da equação acima, é possível notar que seis operadores de Euler independentes e seis operações inversas são suficientes para manipular a estrutura topológica dos modelos de variedade múltipla. Por conveniência das operações de modelagem, porém, oito operadores de Euler adicionais foram incluídos. Também foram adicionados dois operadores topológicos que, apesar de não estarem diretamente relacionados à fórmula de Euler, permitem gerar ou destruir o esqueleto inicial ou final. O quadro V.2 lista os operadores básicos e os estendidos, a partir dos quais todas as operações de alto nível são implementadas.

OPERADOR	SIGNIFICADO	V	E	F	L	S	C	R
Básicos:								
MEV	Constrói aresta, vértice	+1	+1	0	0	0	0	0
KEV	Destrói aresta, vértice	-1	-1	0	0	0	0	0
MEC	Constrói aresta, ciclo	0	+1	0	0	0	+1	0
KEC	Destrói aresta, ciclo	0	-1	0	0	0	-1	0
MFKC	Constrói face, destrói ciclo	0	0	+1	0	0	-1	0
KFMC	Destrói face, constrói ciclo	0	0	-1	0	0	+1	0
MFR	Constrói face, região	0	0	+1	0	0	0	+1
KFR	Destrói face, região	0	0	-1	0	0	0	-1
MVS	Constrói vértice, casca	+1	0	0	0	+1	0	0
KVS	Destrói vértice, casca	-1	0	0	0	-1	0	0
MVL	Constrói vértice, ciclo	+1	0	0	+1	0	0	0
KVL	Destrói vértice, ciclo	-1	0	0	-1	0	0	0
Adicionais								
SEMV	Divide aresta, constrói vértice	1	1	0	0	0	0	0
JEKV	Une aresta, destrói vértice	-1	-1	0	0	0	0	0
MEF	Constrói aresta, face	0	1	0	0	1	0	0
KEF	Destrói aresta, face	0	-1	0	0	-1	0	0
KEML	Destrói aresta, constrói ciclo	0	-1	0	1	0	0	0
MEKL	Constrói aresta, destrói ciclo	0	1	0	-1	0	0	0
KEMS	Destrói aresta, constrói casca	0	-1	0	0	1	0	0
MEKS	Constrói aresta, destrói casca	0	1	0	0	-1	0	0
Topológicos								
MMR	Constrói modelo, região							
KMR	Destrói modelo, região							

Quadro V.2 – Operadores de Euler para a construção de sólidos de variedade múltipla

V.1.4 – A VALIDAÇÃO DE MODELOS B-REP

Representações B-rep válidas são difíceis e tediosas de serem construídas manualmente, devido à necessidade de satisfazerem a condições métricas e combinatoriais [Req80]. A integridade topológica de um modelo por fronteira impõe restrições a seus elementos visando assegurar sua validade. Exemplificando, a orientação das arestas no ciclo deve ser consistente em todo o modelo: ciclos externos podem ser especificados como orientados no sentido anti-horário, vistos de fora do sólido, enquanto anéis podem ser orientados em sentido horário.

Não só a topologia deve fazer parte do processo de validação, a geometria também deve ser completa e coerente. Isso implica que: associada a toda face, aresta ou vértice deve existir, respectivamente, uma superfície, uma curva ou um ponto; todas as curvas e pontos devem estar sobre a superfície da face que os contém; todos os pontos – exceto vértices isolados – devem estar sobre as curvas apropriadas; tanto uma face quanto o objeto como um todo não devem possuir auto-interseção. A integridade geométrica só poderá ser alcançada se o formato atribuído às faces for coerente com a informação topológica (ex.: as faces devem interceptar-se somente em arestas ou vértices comuns.

A verificação completa de validade não é facilmente fornecida para um modelo B-rep: operadores de Euler são eficientes para assegurar a validade topológica, mas a validade geométrica ainda requer verificações computacionalmente caras. Os maiores problemas de confiabilidade em modeladores B-rep estão relacionados com a geometria e não com a topologia. Problemas de tolerância no cálculo de interseção de curvas podem induzir a resultados errôneos quando uma situação real está próxima ao caso limite. Nesta situação, é possível que a topologia do objeto não concorde com a geometria, gerando resultados totalmente errôneos. A fim de evitar problemas desse tipo, sistemas que utilizam a abordagem por fronteira normalmente a constroem a partir de outra representação, baseando-se em cálculos geométricos e algoritmos de conversão, como é o caso deste modelador, que gera a representação B-rep a partir da avaliação da fronteira da estrutura CSG.

V.1.5 – INCLUSÃO DE SUPERFÍCIES CURVAS

A aproximação de modelos curvos por facetas poligonais planares é adequada para aplicações que não requerem muita precisão numérica, uma vez que permite testar a efetividade do modelo sem levar a cálculos matemáticos complicados. Ela não é adequada, porém, para tarefas que necessitam de uma modelagem precisa e exata de superfícies curvas. Se por um lado a subdivisão em facetas muito finas melhora significativamente a aproximação de objetos curvos, por outro resulta

em uma representação volumosa e complexa, que pode consumir mais tempo de processamento do que o cálculo da interseção entre as superfícies exatas. Além do mais, a representação aproximada pode levar a resultados topologicamente errôneos, quando comparados a resultados exatos, devido à perda de informação decorrente da aproximação, como a ilustrada na figura V.6.

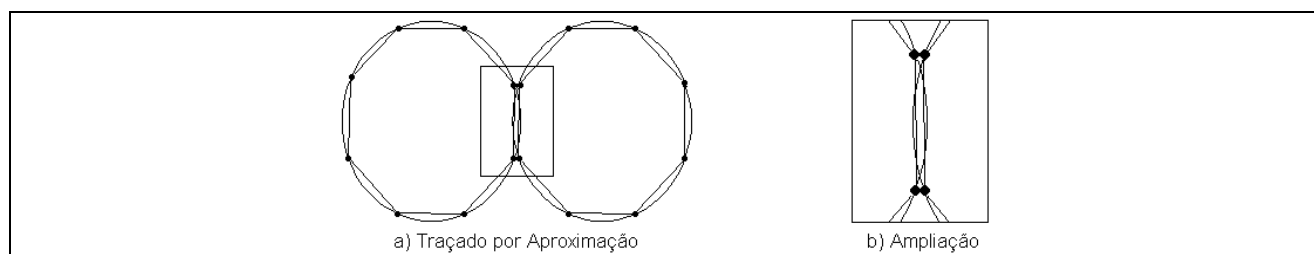



Figura V.6 – Perda de informação devido a aproximação

Sistemas de modelagem que mantêm a geometria exata atingem um nível mais alto de precisão, mas introduzem uma série de complicações topológicas, geométricas e numéricas [Tur88]. Para muitas aplicações, o conjunto de superfícies e curvas que um sistema admite influencia não só o conjunto de objetos que podem ser por ele modelados, mas também o seu desempenho. O uso de vários tipos de geometria em um modelador de sólidos possui a vantagem de possibilitar a representação, de maneira exata, de curvas e superfícies comuns, como círculos, seções cônicas, cilindros e esferas. Sua desvantagem está na dificuldade de adicionar um novo tipo de curva ou superfície ao modelador, uma vez que será necessário desenvolver um número significativo de rotinas de interseção entre o novo tipo de curva e as já existentes. Este é o motivo pelo qual muitos projetistas de modeladores optaram pela geometria por facetas ou pela inclusão de somente um tipo de superfície e curva [Mor85, Rog90]. No desenvolvimento deste modelador, optou-se pela utilização de modelos poliedrais utilizando a geometria por facetas, uma vez que a malha volumétrica a ser obtida será composta por elementos finitos de primeira ordem.

V.1.6 – O PROCESSO DE AVALIAÇÃO DA FRONTEIRA

A visualização e a verificação de um objeto CSG com propósitos de análise é uma tarefa difícil, uma vez que a árvore CSG armazena pouca informação sobre o objeto – basicamente transformações geométricas e operações booleanas sobre primitivas. Torna-se necessário, portanto, realizar uma conversão para a representação de sua fronteira, de modo a permitir uma descrição do modelo baseada em suas faces, arestas e vértices, definindo sua superfície limitante. O processo que realiza essa conversão é denominado "avaliação da fronteira".

Várias implementações para a avaliação da fronteira apresentadas na literatura estão baseadas em sólidos poliedrais [Män83, Req85, Män86, Pil89, Gur91, Muu91], com diferentes visões quanto à importância das informações sobre conectividade e organização dos cálculos. As principais distinções estão no esquema de representação utilizado (B-rep puro ou dual CSG⇔B-rep), no conjunto de sólidos manipulados (de variedade bidimensional ou múltipla), no que é classificado (faces ou arestas) e em como a classificação é realizada (sobre qual representação, em que seqüência, com registro implícito ou explícito das relações de adjacência). Os algoritmos parecem ser um pouco diferentes, mas realizam as mesmas operações essenciais e podem ser aplicados para *r-sets* em geral.

A Avaliação da Fronteira envolve a produção da representação B-rep a partir de modelos CSG, realizando a conversão entre representações. Pode ocorrer de forma não incremental – que obtém apenas a representação B-rep final do modelo CSG, não gerando resultados intermediários – ou incremental – que obtém a representação da fronteira resultante de cada operação boolena aplicada no modelo CSG, seguindo a estratégia "dividir-para-conquistar". Este trabalho segue a abordagem incremental, manipulando resultados intermediários – fronteiras de sub-árvores – e permitindo "rastrear" os sólidos definidos. Um avaliador de fronteira incremental pode ser aplicado recursivamente sobre uma árvore CSG para produzir representações B-rep para a raiz e para todas as suas sub-árvores. Para garantir a obtenção de sólidos manufaturáveis, é essencial que os sólidos modelados sejam algebricamente fechados em relação às operações booleanas, assegurando que os resultados possam ser usados como entradas para outras operações [Til80]. Os operadores booleanos padrões não preservam a solidez, devendo portanto ser substituídos por versões regularizadas que garantam o fechamento, como as apresentadas na seção IV.3.5.

V.2 – Principais Características de Projeto

O projeto do Subsistema de Representação envolveu a definição das estruturas de dados auxiliares, o modelamento e o processo de construção da estrutura B-rep a ser incorporada ao modelador. A especificação de requisitos procurou identificar as principais características a serem incluídas na representação B-rep, o que facilitou sua definição. A partir da especificação foram relacionadas as principais classes envolvidas com a representação e construção de sólidos que, agrupadas, determinaram os assuntos a serem tratados. A seguir, procurou-se obter soluções orientadas para objetos a serem adotadas pela implementação.

V.2.1 – Requisitos Relacionados à Representação Interna ou Núcleo

O Subsistema de Representação, ou Núcleo, é responsável pelo gerenciamento e acesso direto às principais estruturas de dados: qualquer acesso a representações de objetos exigido pelos outros subsistemas, seja para criação, alteração ou armazenamento, é feito por este gerenciador, que constitui o núcleo do modelador. As informações presentes nas estruturas de dados poderão ser obtidas por métodos de outros subsistemas, mas deverão ser criadas e modificadas apenas por métodos pertencentes a classes do Núcleo. O Subsistema de Representação deve cuidar de todos os acessos solicitados pelos outros subsistemas, de forma a garantir a integridade das estruturas de dados. Também fica a cargo deste subsistema acionar mecanismos para armazenar descrições dos objetos em bases de dados permanentes, bem como fornecer à Interface rotinas para preparo dos dados disponíveis, visando à comunicação com outros modeladores e aplicações externas.

Além de possuir os elementos de fronteira usuais em modeladores B-rep, a estrutura de dados B-rep a ser utilizada por este modelador deverá combinar a garantia de geração de modelos manufaturáveis, fornecida pela estrutura Semi-aresta, com a flexibilidade presente na estrutura Aresta-radial, permitindo que várias faces compartilhem uma mesma aresta. O interesse pelo desenvolvimento de uma nova estrutura surgiu da necessidade de analisar a interface entre meios, representada geometricamente pelas fronteiras existentes entre os sólidos que compõem um modelo. Apesar de cada componente possuir sua própria fronteira, para a simulação eletromagnética do modelo é necessário analisar o contato existente entre a sua fronteira e a de outro componente. Essa fronteira de contato precisa ser representada de maneira única, de forma a garantir a compatibilidade da malha de elementos finitos a ser posteriormente gerada.

A representação B-rep deverá armazenar não só as informações topológicas e geométricas sobre os objetos, mas também os atributos adicionais relacionados às características físicas do modelo e do problema a ser resolvido – como material, cor, potencial, corrente, tensão, enrolamentos, condições de contorno, etc.

A representação B-rep só poderá ser manipulada por meio de algoritmos básicos de acesso que implementam os operadores de Euler e alguns procedimentos geométricos elementares. Os operadores de Euler garantem a integridade topológica dos objetos gerados e fornecem a base para a implementação dos mecanismos que desfazem operações errôneas ou indesejadas. Uma base de dados auxiliar deverá armazenar a seqüência de operadores de Euler utilizados e seus parâmetros, permitindo a construção dos objetos pelo Subsistema de Representação.

V.2.2 – A ESTRUTURA B-REP DEFINIDA PARA ESTE MODELADOR

A estrutura de dados desenvolvida para este modelador utiliza o conceito de *s-sets*, apresentado no item IV.3.6.1, acoplado aos conceitos de semi-aresta e de aresta radial. Uma vez que a estrutura semi-aresta representa de forma robusta e completa todos os sólidos de variedade bidimensional, e a potência descritiva pretendida neste modelador envolve a representação específica destes sólidos "montados" entre si, a idéia básica da estrutura apresentada está fundamentada no princípio de que cada região delimitada deverá ser representada por uma estrutura de dados que permita manipular *apenas* sólidos de variedade bidimensional – a estrutura semi-aresta –, que deve ser acoplada a cada região em separado. Assim, ao invés de trabalhar com semi-arestas sobre as arestas do modelo, pode-se trabalhar com um par de semi-arestas para cada *uso* de aresta (ou seja, uso-aresta), associando uma semi-aresta a cada um de seus vértices limitantes (os uso-vértices) e definindo uma orientação para as faces da região (os uso-faces). A figura V.7 elucida esta idéia.

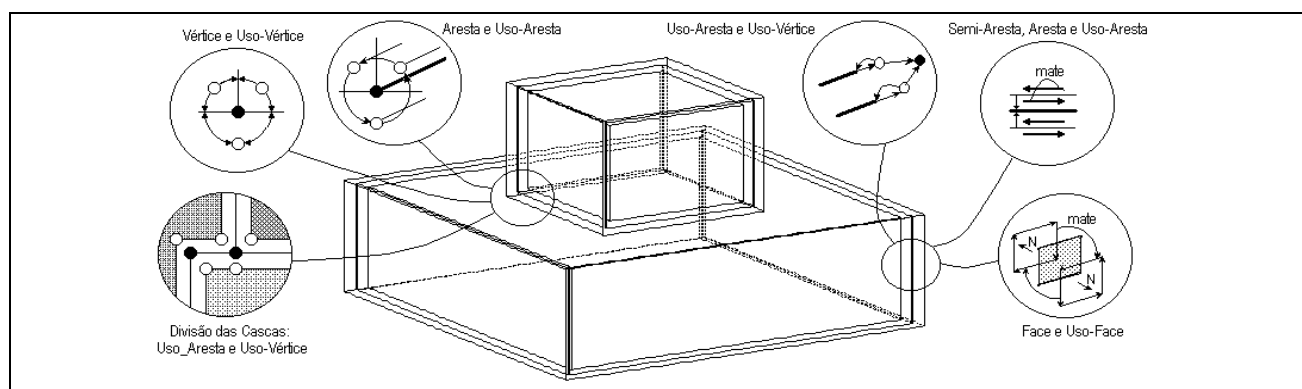



Figura V.7 – Características da estrutura B-rep proposta: uso-face, uso-ciclo, uso-aresta, semi-aresta e uso-vértice

Cada região delimitada possui sua própria orientação, compatível com a orientação dos demais componentes do modelo. Atenção especial deve ser dada à fronteira entre os componentes de uma montagem, como ilustrado na figura V.8: cada face da fronteira possui dois uso-faces – um para cada região –, cuja orientação é definida pelo uso-ciclo externo; buracos na face são definidos por uso-ciclos internos, inversamente orientados; cada ciclo é composto por uma seqüência de semi-arestas, definidas sobre cada uso-aresta, que por sua vez são delimitadas por uso-vértices; as semi-arestas são responsáveis pela orientação de todo o modelo e são sempre tratadas aos pares (*mate*).

No contexto de montagens, o conceito de Aresta-radial continua sendo muito importante, por permitir que várias faces compartilhem a mesma aresta, o que é fundamental na definição de fronteiras internas. Além de desnecessária, porém, a representação de elementos de menor dimensão – como vértices, arestas ou faces – soltos ou pendentes não é conveniente, e só se justifica durante a

construção de componentes sólidos. A fim de representar exclusivamente sólidos manufaturáveis e garantir a validade do modelo, a parte da estrutura de dados referente a sólidos de variedade múltipla pode ser desconsiderada, e uma outra estrutura que permita manipular sólidos de variedade bidimensional orientados deve ser incluída.

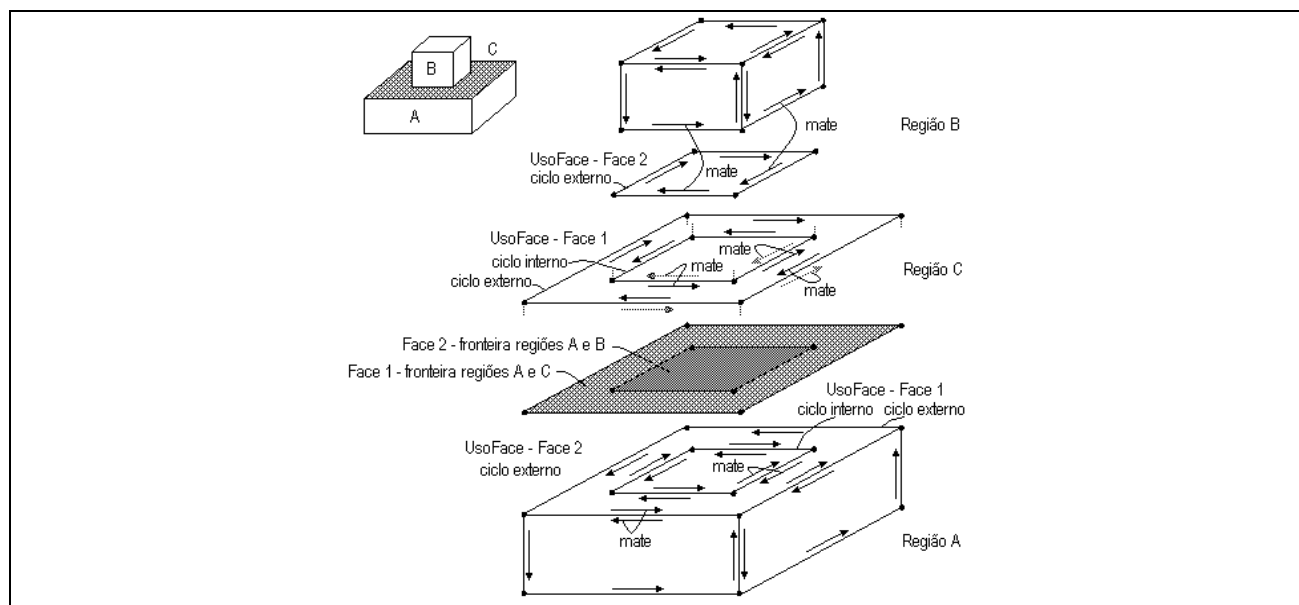



Figura V.8 – Elucidação do conceito de uso-face e seus derivados

V.2.3 – Operadores de Euler a serem Utilizados pela Estrutura B-rep Definida

Uma vez definida a estrutura de dados B-rep a ser utilizada, deve-se estabelecer o conjunto de operadores de Euler que permitirá manipulá-la corretamente. Para isso é necessário rever a fórmula de Euler e redefinir seus operadores. Como o interesse deste trabalho está no acoplamento de componentes de variedade bidimensional, parece interessante partir da fórmula apresentada para sólidos de variedade bidimensional e incluir nela a noção de *s-sets* e de compartilhamento de faces. Arbab [Arb90] não apresenta um conjunto de operadores de Euler para *s-sets*, mas Wilson [Wil85] descreve variações nas fórmulas de Euler para diversas situações. Segundo Wilson, a fórmula de Euler poderia ser escrita como:

$V - E + 2F - L - (S - G) = 0$ para sólidos de variedade bidimensional abertos

$V - E + 2F - L - 2(S - G) = 0$ para sólidos de variedade bidimensional fechados

sendo que V, E e F correspondem respectivamente ao número de vértices, arestas e faces do sólido, L representa o número total de ciclos – externos e internos – em faces, S é o número de cascas e G o *genus* – número de buracos que atravessam o sólido. Pode-se observar que: para cada face existe

somente um ciclo externo, ou seja, o número de ciclos externos é igual ao número de faces; os ciclos internos correspondem aos anéis (cavidades em faces), denominado anteriormente de R; G é o mesmo que o H anterior. Assim, as fórmulas $V - E + 2F - L - 2(S - G) = 0$ e $V - E + F = 2(S - H) + R$ dizem exatamente a mesma coisa.

Wilson afirma ainda que objetos abertos podem ser tratados pela fórmula para objetos fechados, desde que seja incluída a noção de "*dummy face*" – face adicional, composta por todas as arestas de contorno do objeto aberto, que o converte em um objeto fechado. A adição de uma "*dummy face*" a um objeto aberto aumenta não só a contagem de faces, mas também o número de ciclos do objeto pela geração de "*dummy loops*", mantendo a equação anterior válida. O número de "*dummy loops*" pode ser determinado "removendo" todas as arestas do objeto aberto, menos as arestas do contorno, e contando os ciclos formados pelas arestas de contorno, como exemplificado na figura V.9.

Figura V.9 – Alguns exemplos de contagem de elementos [Arb90]

Manipulando as duas fórmulas, com inclusão da contagem adicional de "*dummy face*" e "*dummy loops*", obtém-se:

$V' = V$; $E' = E$; $F' = F + dF$; $L' = L + dL$; $S' = S$; $G' = G$

$V - E + 2F - L - (S - G) = V' - E' + 2F' - L' - 2(S' - G')$

$V - E + 2F - L - (S - G) = V - E + 2(F + dF) - (L + dL) - 2(S - G)$

$V - E + 2F - L - S + G = V - E + 2F + 2dF - L - dL - 2S + 2G \Rightarrow 0 = 2dF - dL - S + G$

Como $dL = G + 1$, obtém-se $2dF - G - 1 - S + G = 0 \Rightarrow 2dF = S + 1$

Isto sugere que é possível dividir uma casca em duas outras que compartilham entre si a fronteira, bastando acrescentar duas "*dummy faces*", como mostra a figura V.10, mantendo ainda a validade topológica do modelo por continuar satisfazendo a fórmula de Euler. Cada uma destas cascas geradas pode ser novamente dividida, e o resultado obtido será sempre um conjunto de sólidos de variedade bidimensional que fazem fronteiras entre si. Essas "*dummy faces*" correspondem aos usos da face que está sendo compartilhada pelas duas cascas, e o conceito de uso-face, já incorporado à estrutura de dados, pode ser também incorporado aos operadores de Euler. Duas novas operações podem então ser definidas, uma para construir e outra para destruir esse tipo de fronteira.

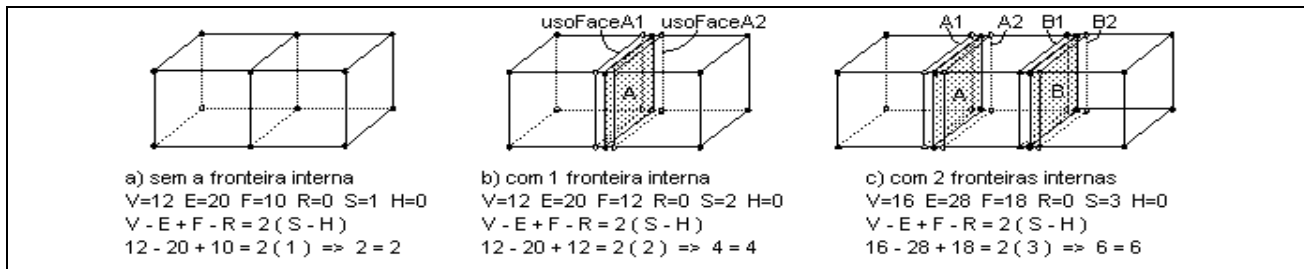



Figura V.10 – Exemplos de divisão de cascas utilizando o compartilhamento de fronteiras

Conseqüentemente, ainda é possível extrapolar os exemplos bidimensionais da figura V.9 para situações tridimensionais, ou seja, simular uma varredura de modo a transformar vértices em arestas, arestas em faces e faces em volumes, obtendo-se conjuntos abertos semelhantes aos *s-sets*. Sobre eles poderia ser aplicada a fórmula para sólidos de variedade múltipla ou a fórmula para conjuntos abertos, considerando, obviamente, valores corretos para as variáveis dentro de cada contexto. A fórmula para conjuntos fechados também poderia ser aplicada da mesma forma, desde que fosse considerada a contagem de “*dummy faces*” e de “*dummy loops*”, mas sendo agora tratada como “*dummy shell*” – casca adicional, composta por todas as faces de contorno do objeto aberto, que o converte em um objeto fechado – e “*dummy holes*” – buracos no sólido, que são determinados "removendo-se" todas as faces do objeto aberto, menos as faces do contorno, e "contando-se" os buracos formados pelas faces do contorno. Todas as estruturas do tipo “*dummy*” que estão sendo geradas correspondem ao compartilhamento de elementos da fronteira, associados ao conceito de *uso*: uma região estaria delimitada por uma casca externa – a "*dummy shell*"–, composta por faces e "*dummy faces*" – os usos da face para aquela região –, que por sua vez definem usos de arestas e de vértices, podendo ainda possuir ciclos – "*dummy loops*".

Uma vez que a estrutura proposta pode ser vista como uma montagem de sólidos de variedade bidimensional construídos sobre os usos dos elementos face, ciclo, aresta e vértice, cada componente desta montagem satisfaz a fórmula de Euler $V – E + F = 2 (S – H) + R$. Falta, porém, incluir na fórmula o conceito de região, correspondendo à casca mais externa do sólido. A letra *R*, atribuída até então para *Ring* – anéis em faces – pode ser substituída por *L*, de *Loop* – ciclos internos. Como o *S* presente na fórmula anterior referencia o número total de cascas – externa e internas –, ao incluir *R* para Região, o novo *S* referencia apenas cascas internas. Assim, a nova fórmula é:

$V – E + F – L = 2 (R + S – H)$ sendo que:

V, *E* e *F* representam respectivamente vértices, arestas e faces

L é o número de ciclos internos em faces

R é o número de regiões , que corresponde ao número de cascas externas

S é o número de cascas internas, isto é, o número de buracos em regiões

H é o *genus*, ou seja, o número de buracos que atravessam o sólido

Os operadores de Euler a serem utilizados no contexto deste trabalho são apresentados no quadro V.3, e uma descrição mais detalhada de cada operador é apresentada na seção V.3.3. Abrangem basicamente os definidos para sólidos de variedade simples – que geram cada componente do modelo – somados aos que criam fronteiras internas. Apesar de serem utilizados os nomes face, ciclo, aresta e vértice na nomenclatura e descrição, os operadores na realidade serão aplicados sobre os usos destes elementos. A omissão da palavra "uso" foi proposital, visando facilitar a compreensão dos operadores.

OPERADOR	SIGNIFICADO	V	E	F	L	R	S	H
MVFR	*Make Vertex, Face, Region* Constrói vértice, face, região	+1	0	+1	0	+1	0	0
KVFR	*Kill Vertex, Face, Region* Destrói vértice, face, região	-1	0	-1	0	-1	0	0
MEV	*Make Edge, Vertex* Constrói aresta, vértice	+1	+1	0	0	0	0	0
KEV	*Kill Edge Vertex* Destrói aresta, vértice	-1	-1	0	0	0	0	0
MEF	*Make Edge, Face* Constrói aresta, face	0	+1	+1	0	0	0	0
KEF	*Kill Edge Face* Destrói aresta, face	0	-1	-1	0	0	0	0
MEKL	*Make Edge, Kill Loop* Constrói aresta, destrói ciclo	0	+1	0	-1	0	0	0
KEML	*Kill Edge, Make Loop* Destrói aresta, constrói ciclo	0	-1	0	+1	0	0	0
MFKLH	*Make Face, Kill Loop Hole* Constrói face, destrói ciclo e cavidade	0	0	+1	-1	0	0	-1
KFMLH	*Kill Face Make Loop Hole* Destrói face, constrói ciclo e cavidade	0	0	-1	+1	0	0	+1
MSKR	*Make Shell, Kill Region* Constrói casca, destrói região	0	0	0	0	-1	+1	0
KSMR	*Kill Shell, Make Region* Destrói casca, constrói região	0	0	0	0	+1	-1	0
MFFR	*Make double Face, Region* Constrói face dupla, região	0	0	+2	0	+1	0	0
KFFR	*Kill double Face, Region* Destrói face dupla, região	0	0	-2	0	-1	0	0
AR	*Assemble regions* Junta duas regiões por duas faces iguais	-nV	-nV	0	0	+1	0	+1
DR	*Disassemble regions* Separa duas regiões contendo face em comum	-nV	-nV	0	0	-1	0	-1
AF	*Assemble faces* Após AR, junta duas outras faces iguais	-nV	-nV	0	0	+1	0	+1
DF	*Disassemble face* Separa uma face em duas, antes de DR	-nV	-nV	0	0	-1	0	-1
GR	*Glue Regions* Cola duas regiões por duas faces iguais	-nV	-nV	-2	0	-1	0	0
UR	*Unglue Region* Descola região por uma face	-nV	-nV	+2	0	+1	0	0
GF	*Glue faces* Após UR, cola duas outras faces iguais	-nV	-nV	-2	0	-1	0	0
UF	*Unglue face* Descola uma face -vira duas– aplica antes de UR	+nV	+nV	+2	0	+1	0	0

Quadro V.3 – Operadores de Euler a serem utilizados nesse trabalho

V.2.4 – O MODELAMENTO OBTIDO

A especificação de requisitos do Núcleo possibilitou identificar as principais classes e as relações de dependência entre elas, conforme ilustrado na figura V.11. Uma análise um pouco mais detalhada permitiu reconhecer os atributos e métodos essenciais de cada classe e com o resultado obtido, esquematizado no quadro V.4, foi possível identificar os assuntos tratados pelo Subsistema de Representação, bem como relacioná-los entre si e com os assuntos dos outros subsistemas. A estruturação de assuntos obtida é apresentada na figura V.12 e descrita a seguir.

As Estruturas de Dados Auxiliares gerenciam a criação e a manipulação de estruturas de dados específicas do modelador, entre elas: vetores e matrizes contendo diversos tipos de dados e utilizados por todo o sistema; listas para armazenar primitivas planares e sólidas, definidas pelo usuário e utilizadas pelo Subsistema de Modelagem; listas para armazenar parâmetros de um comando bem como registrar comandos executados; a lista básica a ser utilizada pelos elementos que compõem a hierarquia existente na estrutura B-rep.

A Avaliação da Fronteira envolve as classes e métodos que realizam a conversão entre as representações CSG e B-rep, ou seja, geram a representação da fronteira do modelo a partir da especificação, pelo usuário, de primitivas e operações de modelagem a serem realizadas. A avaliação da fronteira de cada primitiva é inicialmente realizada. Novas faces, arestas e vértices são gerados sobre a fronteira das primitivas, caso exista interseção entre elas. Todos os elementos da fronteira são então classificados quanto à sua posição em relação aos operandos, e em função da operação aplicada, são selecionados como parte de seu resultado. Desta forma, obtêm-se as fronteiras de sub-árvores CSG e aplica-se recursivamente o processo de avaliação da fronteira até que a representação B-rep para a raiz da árvore seja obtida.

As Operações de Euler englobam todos os operadores responsáveis pela construção de modelos B-rep topologicamente válidos. Atuam diretamente nas primitivas sólidas definidas pela representação CSG, determinando sua fronteira e, indiretamente, nas operações de modelagem especificadas, devido à criação e destruição de elementos B-rep realizadas pela Avaliação da Fronteira.

A Representação B-rep contém a definição das classes relacionadas com a construção da fronteira do modelo a partir dos Operadores de Euler. Manipula elementos geométricos básicos – pontos, retas e faces – e determina as superfícies sobre as quais o Subsistema de Geração de Malha construirá a malha superficial. Também fornece todos os dados geométricos necessários para a criação da base de dados neutra.

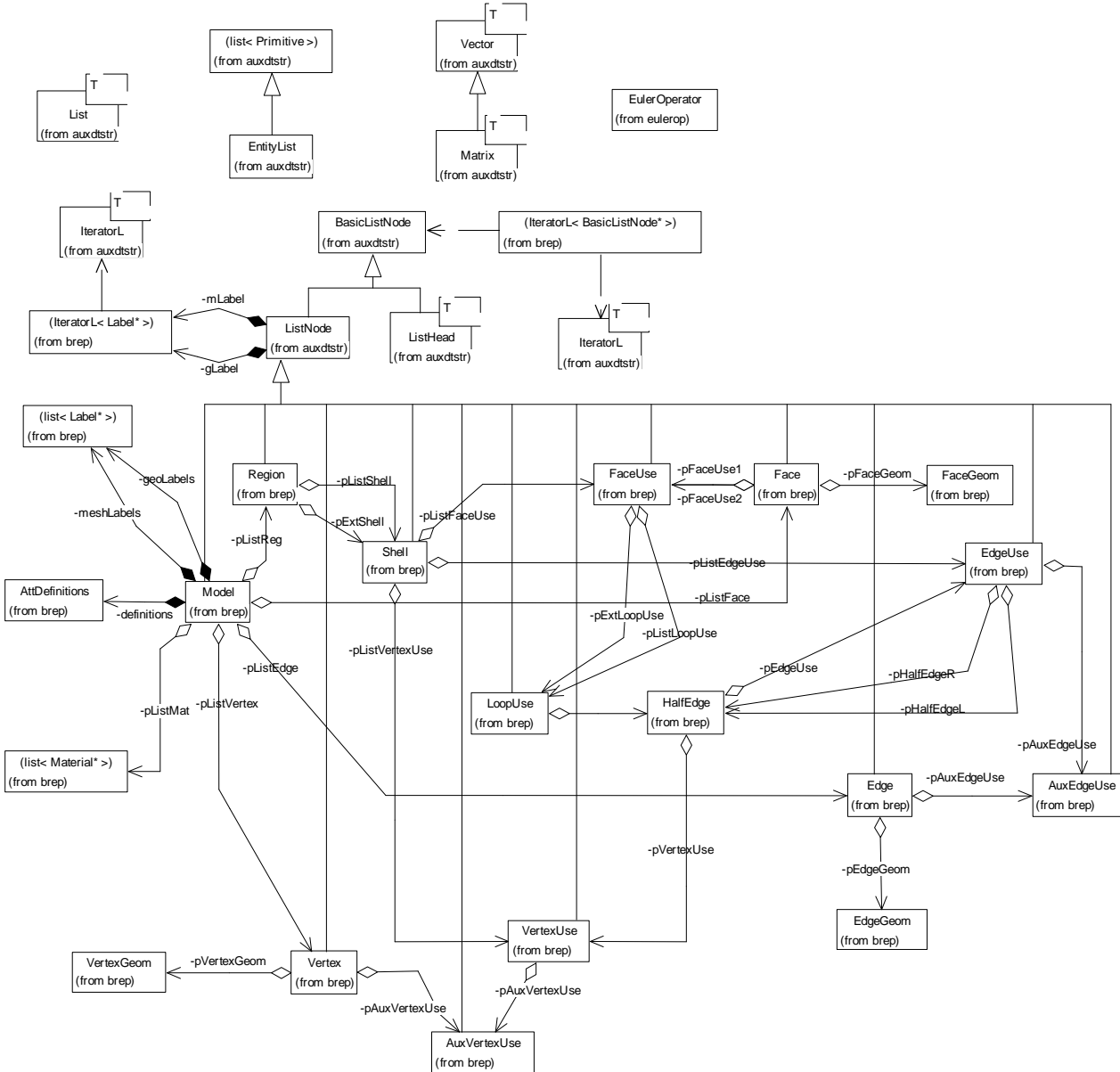



Figura V.11 – Diagrama das principais classes e relações de dependência entre elas

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
BasicListNode: classe abstrata, definindo o nó básico da lista a ser utilizada pelos componentes da estrutura B-rep	
Nenhum	Homogeneizar o tratamento do nó-cabeça e do nó comum da lista, obter nó, verificar se é nó cabeça
ListHead: nó-cabeça da lista a ser utilizada pelos componentes da estrutura B-rep	
Referência ao nó-pai – um nó do tipo ListNode de uma lista de componentes B-rep hierarquicamente superior	Evitar que todos os nós da lista possuam esta mesma referência, setar e recuperar referência
ListNode: nó comum da lista a ser utilizada pelos componentes da estrutura B-rep	
Nenhum	Homogeneizar tratamento dos componentes B-rep, mostrar conteúdo, setar referência ao pai
List: lista-gabarito a ser utilizada pelos componentes da estrutura B-rep; contém um nó-cabeça diferente dos demais nós	
Nenhum	Todos os prestados por uma lista padrão STL (posicionar no início e fim, obter anterior ou próximo, localizar nó, incluir no final, inverter ordem, obter tamanho) e setar e recuperar pai
IteratorL: agente de interação da lista a ser utilizada pelos componentes da estrutura B-rep	
Nenhum	Corrigir as diferenças de caminhamento na lista devido ao nó-cabeça e aceitar valor nulo
Vector: vetor com alocação dinâmica, redefinido sobre o vetor da STL para incluir funções de interesse do modelador	
Nenhum	Todos os prestados por um vetor STL, mais incluir elemento no início
Matrix: matriz com alocação dinâmica, definida como um vetor de vetores pertencentes à classe Vector	
Nenhum	Obter identidade, transposta, calcular determinante, cofator, resolver sistemas por Cramer, incluir novos elementos na linha ou coluna, tamanho
Model: modelo que está sendo construído	
Identificador do modelo; nome; referência para as listas de regiões, faces, arestas, vértices; próximo número disponível e quantidade de regiões, cascas, faces, arestas, vértices, usoFaces, usoArestas, usoVértices; referência para a lista de labels de geometria e malha do modelo	Obter identificador, nome, lista de *labels* de geometria e malha, tabelas de atributos, listas de regiões, arestas, faces, vértices; adicionar e remover região, aresta, face, vértice; acertar e obter próximo número e quantidade dos componentes; criar arquivo ".3D"; apagar de forma consistente todas as suas referências
Region: uma das regiões que compõem o modelo	
Identificador da região, nome, referência para a casca externa, referência para a lista de cascas da região, referência ao label associado obs: o nó-cabeça da lista de regiões referencia o modelo que contém a região	Obter identificador e nome; acertar e obter casca externa; adicionar, obter e subtrair casca; apagar de forma consistente todas as suas referências; atribuir e obter referência às cascas
Shell: uma das cascas que compõem a região	
Identificador da casca, referência para as listas de usoFaces, usoArestas e usoVértices, referência ao label associado à casca obs: o nó-cabeça da lista de cascas referencia a região que contém a casca	Obter identificador, adicionar, obter e subtrair usoFace, usoAresta e usoVértice; apagar de forma consistente todas as suas referências; atribuir e obter referência às usoFaces
Face: uma das faces do modelo	
Identificador da face, referência para as duas ocorrências de usos da face, referência para a definição da geometria da face, referência ao label associado à face.	Obter identificador; atribuir, obter e apagar geometria; atribuir e obter referência às usoFaces
FaceUse: um dos usos de face que compõem a casca	
Identificador da usoFace, referência para o usoCiclo externo da usoFace, referência para a lista de usoCiclos da usoFace, referência para a face à qual este usoFace se refere, orientação da normal em relação à fàce (+ / –) obs: o nó-cabeça da lista de usoFaces referencia a casca que contém a usoFace	Obter identificador; acertar, obter e remover uso-Ciclos; acertar e obter face e usoCiclo externo; obter semi-aresta; solicitar acerto e obtenção de limites; apagar de forma consistente todas as suas referências

Quadro V.4 – Principais características das classes relacionadas (início)

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
FaceGeom: descrição da geometria associada à face	
Plano (superfície, no futuro) que contém a face, limites mínimo e máximo do envelope convexo que contém a face	Acertar e obter limites, acertar e obter plano (superfície) que contém a face
LoopUse: um dos usoCiclos que compõem a face	
Referência para a lista de semi-arestas do usoCiclo obs: o nó cabeça da lista de usoCiclos referencia a usoFace que contém o usoCiclo	Obter identificador; adicionar, obter e remover semi-arestas; acertar e obter lista de semi-aresta; acertar referência da lista de semi-arestas para si
Edge: uma das arestas do modelo	
Identificador da aresta, referência para a lista auxiliar de usos da aresta, referência para a definição da geometria da aresta, referência ao *label* associado à aresta	Adicionar, remover e obter nó da lista auxiliar de usos da aresta; obter identificador; atribuir, obter e apagar geometria
EdgeUse: um dos usos de uma aresta; um dos usoAresta que compõem a casca	
Identificador da usoAresta, referências para as duas ocorrências de semi-arestas, referência para o seu auxiliar de uso da aresta obs: o nó cabeça da lista de usoArestas referencia a casca que contém a usoAresta	Obter identificador, acertar e obter semi-arestas e auxiliar de uso da aresta, obter aresta, obter casca, obter semi-aresta par de uma fornecida; apagar de forma consistente todas as suas referências
AuxEdgeUse: auxiliar de uso da aresta, que permite relacioná-lo com sua casca e com a aresta do qual se origina	
Referência para o usoAresta pertencente a uma lista de usoArestas de uma casca obs: o nó cabeça da lista de auxUsoArestas referencia a aresta do qual ele é derivado	Associar o usoAresta simultaneamente à sua casca e à aresta da qual é derivado – obtém e define o usoAresta na lista de usoArestas de uma casca, obtém a aresta do usoAresta
EdgeGeom: descrição da geometria associada à aresta	
Reta (curva, no futuro) que contém a aresta, limites mínimo e máximo do envelope convexo que contém a aresta	Acertar e obter limites, acertar e obter reta (curva) que contém a aresta
Vertex: um dos vértices do modelo	
Identificador do vértice, referência para a definição da geometria do vértice, referência para a lista auxiliar de usos do vértice, referência ao *label* associado ao vértice	Adicionar, remover e obter nó da lista auxiliar de usos do vértice; obter identificador; atribuir, obter e apagar geometria
VertexUse: um dos usos de um vértice; um dos usoVértices que compõem a casca	
Identificador do usoVértice, referência para a classe auxiliar que relaciona os usos de um mesmo vértice obs: o nó cabeça da lista de usoVértices referencia a casca que contém o usoVértice	Obter identificador e vértice; obter e acertar o auxiliar de uso do vértice; obter coordenada; apagar de forma consistente todas as suas referências
AuxVertexUse: auxiliar de uso do vértice, que permite relacioná-lo com sua casca e com o vértice do qual se origina	
Referência para o usoVértice pertencente a uma lista de usoVértices de uma casca obs: o nó cabeça da lista de auxUsoVértices referencia o vértice do qual ele é derivado	Associar o usoVértice simultaneamente à sua casca e ao vértice do qual é derivado – obtém e define o usoVértice na lista de usoVértices de uma casca, obtém o vértice do usoVértice
VertexGeom: descrição da geometria associada ao vértice	
Coordenadas do vértice	Acertar e obter coordenadas dos vértices
HalfEdge: uma das semi-arestas associadas a uma aresta, responsável por definir a orientação da face	
Referência para a usoAresta da qual ela faz parte, referência para o usoVértice do qual ela se origina obs: o nó cabeça da lista de semi-arestas referencia o usoCiclo que contém a semi-aresta	Obter identificador do vértice e da aresta; obter aresta, vértice, usoAresta, usoVértice e auxiliares de uso da aresta e do vértice; acertar usoAresta e usoVértice; obter coordenada do vértice onde se origina; obter sua semi-aresta par; apagar de forma consistente todas as suas referências
EulerOperator: operadores de Euler para manipulação da estrutura B-rep	
Nenhum	Construir modelos B-rep utilizando os operadores: MVFR, KVFR, MEV, KEV, MEF, KEF, MEKL, KEML, MFKLH, KFMLH, MSKR, KSMR, MFFR, KFFR, AR, DR, AF, DF, GR, UR, GF, UF
AttDefinitions: contém todas as tabelas de atributos definidas para o modelo	
Todas as tabelas de atributos definidas para montar *labels* (obs: mesma classe já definida no subsistema de interface)	Gerenciar tabelas: adicionar, obter, alterar atributos, associar atributos (*labels*) a elementos do modelo

Quadro V.4 – Principais características das classes relacionadas (continuação)

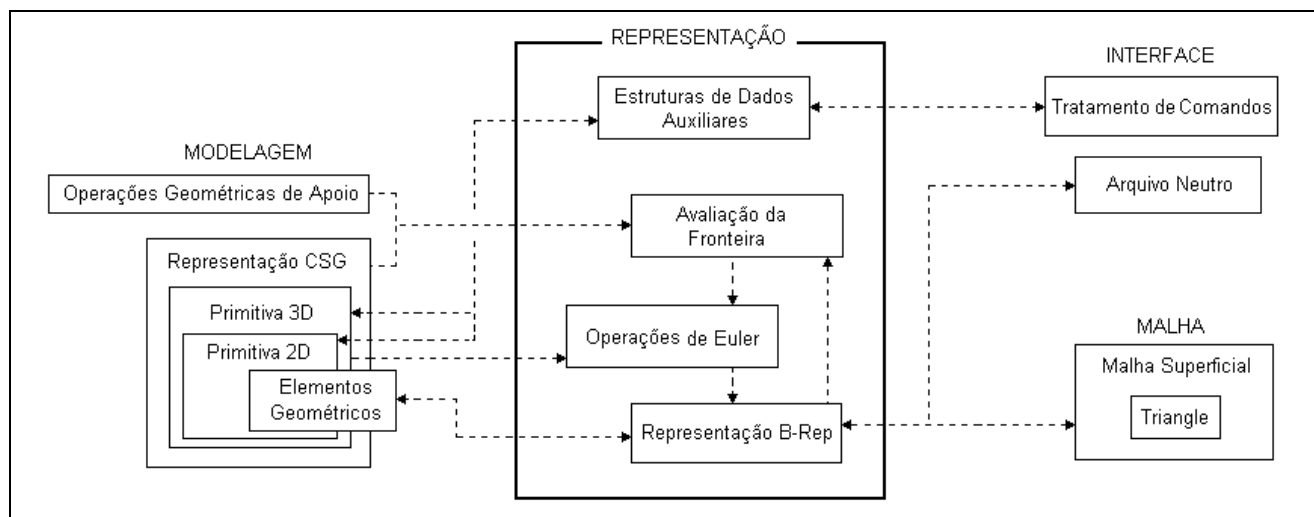



Figura V.12 – Identificação de assuntos relacionados ao Subsistema de Representação

V.2.5 – Soluções Orientadas para Objetos

A hierarquia definida para a estrutura de dados B-rep corresponde aos elementos primitivos modelo, região, casca, face, aresta, semi-aresta e vértice. Para face, aresta e vértice, existem elementos derivados representando suas ocorrências em cada região (usoFace, usoAresta e usoVértice) e suas geometrias (geomFace, geomAresta e geomVértice). A inclusão do elemento ciclo na estrutura possibilitaria a criação de faces com buracos. Como, porém, o conceito de ciclo está diretamente relacionado à orientação adotada pela face, e o que está sendo orientado é o uso da face, nesta estrutura considerou-se correto acrescentar o conceito de usoCiclo relacionado ao de usoFace. O modelo orientado para objetos, apresentado na figura V.13 em notação UML [Boo00], é basicamente composto por estruturas todo-parte e associações. Estruturas auxiliares permitem fazer a ligação entre usos da mesma aresta e usos de arestas na mesma casca, simplificando os relacionamentos de muitos para muitos (m:n) existentes entre casca e vértice. A mesma situação ocorre entre os usos de um mesmo vértice e os usos de vértices em uma casca.

Outra opção de projeto adotada diz respeito à estrutura de dados em lista. Devido principalmente à sua natureza dinâmica, a estrutura B-rep é constituída basicamente por listas lineares duplamente encadeadas. Uma estrutura desse tipo é representada em um computador por uma série de células, denominadas nós, cada uma delas dividida em duas partes: a primeira contém o item da lista – informação sobre o elemento propriamente dito – e a segunda referencia a célula imediatamente anterior e a posterior na lista, como ilustra a figura V.14a. A princípio, entre as informações de cada elemento da estrutura B-rep há uma referência para o elemento hierarquica

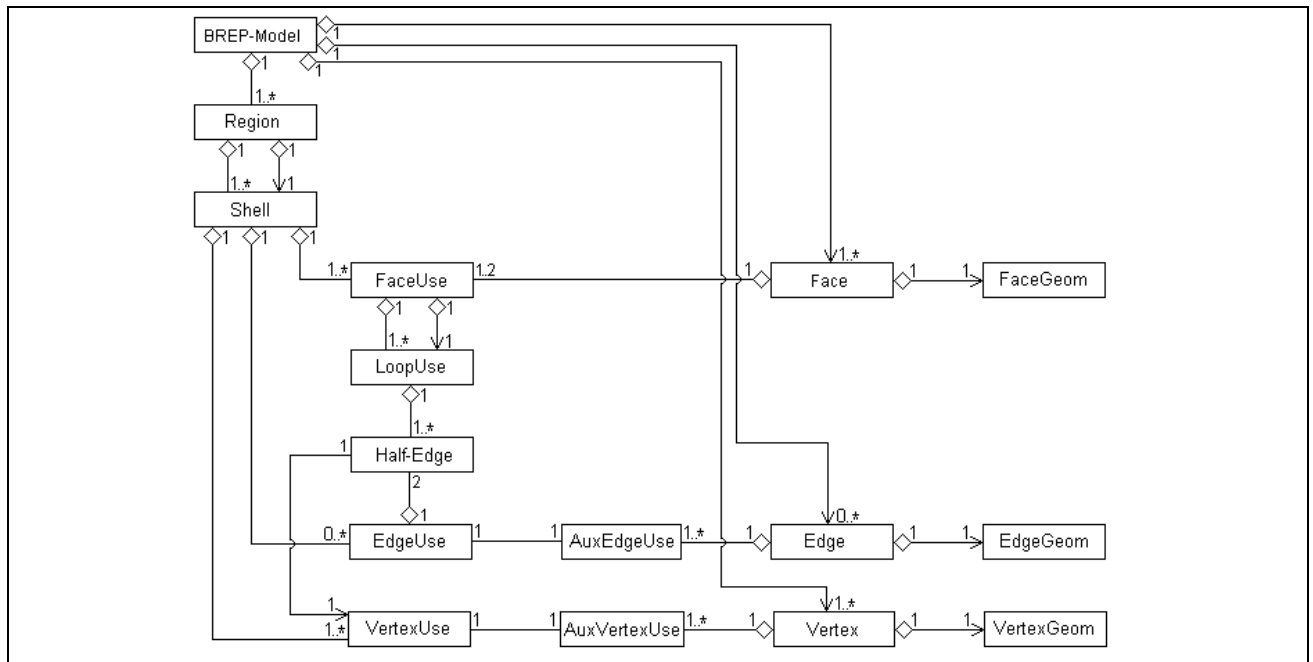



Figura V.13 – Relações de dependência da estrutura *BREP-Model*

mente superior (ex.: um usoAresta referencia um usoFace, que referencia uma casca, que por sua vez referencia uma região pertencente ao modelo). Como todos os elementos de uma lista referenciam o mesmo elemento superior, tornou-se desnecessário e redundante armazenar esta informação repetidamente. Optou-se, portanto, pela criação de uma lista semelhante à ilustrada na figura V.14b, que contém um nó-cabeça cujo conteúdo, diferentemente dos demais nós da lista, é a referência ao elemento hierarquicamente superior. Isto não só economiza memória, mas também garante uma referência consistente, uma vez que impede o armazenamento de diferentes valores pelos elementos da lista. Dependendo do objetivo de cada método ou função a ser executada, o nó-cabeça pode ser ou não incluído no percurso dessa lista: quando é necessário obter a referência ao elemento superior, a lista é percorrida até atingir o nó-cabeça, em caso contrário, ela é percorrida somente entre os elementos de mesma configuração, excluindo o nó-cabeça. Se por um lado a necessidade de percorrer a lista até o nó-cabeça retarda a obtenção da referência ao elemento superior, por outro a existência de uma referência única diminui a necessidade de atualização e verificação da unicidade de seu valor, além de deixar a estrutura mais compacta.

a) lista linear duplamente encadeada convencional b) lista otimizada, implementada pelo modelador

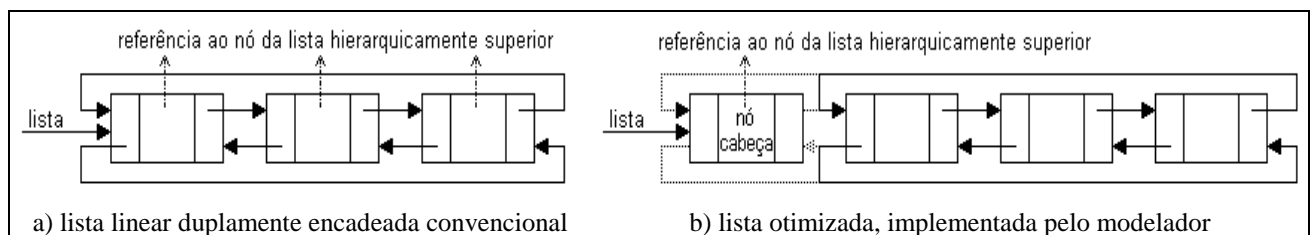



Figura V.14 – A estrutura de lista utilizada pela representação B-rep

V.3 – Principais Características de Implementação

A implementação do Subsistema de Representação teve início com a criação das estruturas de dados auxiliares. Após a construção da lista otimizada, o esqueleto da estrutura B-rep e os operadores de Euler elementares foram criados. À medida que novos componentes eram incluídos na estrutura B-rep, operadores de Euler para manipulá-los eram construídos, até a obtenção de toda a estrutura e dos operadores mais complexos. Nesta etapa, já era possível gerar a representação da fronteira de todas as primitivas CSG definidas e realizar algumas operações de modelagem com objetos pré-posicionados. Finalmente, obteve-se a fronteira resultante da aplicação de operações booleanas ou de montagem, alcançada pela implementação do processo de avaliação da fronteira do modelo CSG.

V.3.1 – Estruturas de Dados Auxiliares

A classe *Vector* implementa vetores contendo um tipo genérico de dado, geralmente denominado gabarito (*template*) ou tipo parametrizado [Per94]. Utiliza alocação dinâmica de memória e foi definida sobre o vetor básico da biblioteca STL (*Standard Template Library*), ao qual foram acrescentados métodos de construção contendo tamanho e valor inicial, além de um método para inserção e retirada de elementos na posição inicial. Derivada da classe *Vector*, a classe *Matrix* implementa matrizes contendo gabaritos alocados dinamicamente na memória. É definida como um vetor de vetores pertencentes à classe *Vector*. Para qualquer tipo de dado, permite incluir novos elementos no final da linha ou coluna, bem como obter seu tamanho. Para valores reais, possui diversos métodos, entre eles: multiplicar matrizes; gerar matriz identidade e transposta; calcular determinante e cofator; resolver sistemas lineares. Essas classes estão definidas no arquivo *Matrix.h*

A classe *EntityList* implementa uma lista de entidades a ser instanciada para armazenar as primitivas 2D e 3D definidas pelo usuário. Utiliza a lista padrão definida pelo STL, que é linear e duplamente encadeada. O conteúdo de cada nó é uma referência a um objeto da classe *Primitive*, que generaliza as classes *Primitiv2D* e *Primitiv3D*. É definida no módulo *EntList*.

O esqueleto da estrutura B-rep foi implementado no módulo *Listc* por meio da classe abstrata *BasicListNode* e das classes concretas *ListHead* e *ListNode*, seguindo a proposta de projeto descrita na seção V.2.4. A classe *BasicListNode* define o nó básico da lista a ser utilizada pelos componentes da estrutura B-rep, homogeneizando o tratamento entre o nó-cabeça – o *ListHead* – e o nó que contém um item da lista propriamente dito – o *ListNode*. O nó-cabeça é o primeiro nó da

lista e contém uma referência ao nó-pai – um nó do tipo *ListNode* pertencente a uma lista de componentes B-rep hierarquicamente superiores. Como essa referência é comum a todos os componentes B-rep pertencentes à lista, sua inclusão no nó-cabeça elimina a necessidade de incluí-la em todos os nós da lista. O *ListNode* é o gabarito utilizado por todos os componentes da estrutura B-rep. A lista de componentes B-rep é definida pela classe *List*, uma lista circular duplamente encadeada composta por *BasicListNode*, implementada a partir da lista padrão definida pela biblioteca STL. Dois agentes de interação são definidos para percorrer essa lista: o *Iterator* padrão do STL, que percorre a lista inteira sem fazer distinção entre os nós, e o *IteratorL*, que percorre somente os nós comuns da lista, excluindo o nó-cabeça. Assim, métodos que necessitam buscar a referência ao componente B-rep hierarquicamente superior utilizam o *Iterator*, enquanto os demais utilizam o *IteratorL*.

V.3.2 – A ESTRUTURA B-REP

A estrutura B-rep definida para esse modelador , ilustrada na figura V.15, está implementada no módulo *Brep*. Toda a estrutura topológica está baseada em listas circulares duplamente encadeadas, como relatado na seção V.2.4. Uma descrição sucinta de seus principais componentes é apresentada no quadro V.5. Os elementos primitivos face, aresta e vértice são fundamentais nesta estrutura: sobre eles ocorrerá a geração da malha de elementos finitos superficial e volumétrica, o que garantirá a compatibilidade da malha gerada na fronteira entre regiões. A estrutura B-rep é construída utilizando os operadores de Euler, cuja interação é controlada pelos métodos definidos no módulo *Cbrep*.

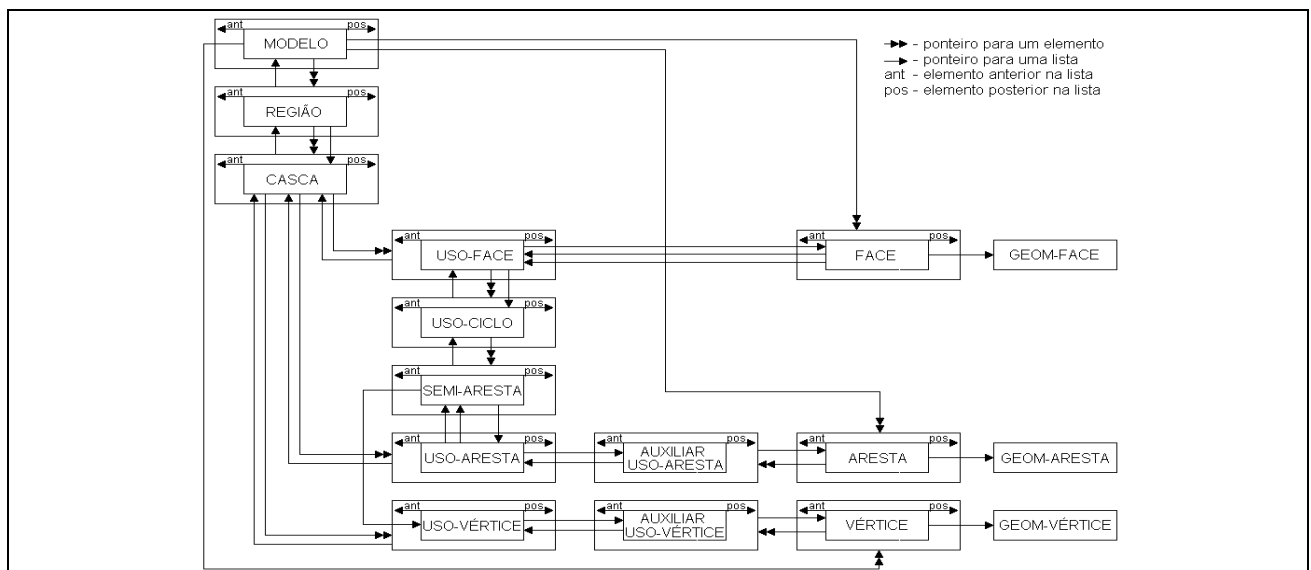



Figura V.15 – A estrutura de dados B-rep implementada nesse trabalho

MODELO
- nome e código identificador do modelo
- apontador para a lista de regiões do modelo
- apontador para a lista de faces do modelo
- apontador para a lista de arestas do modelo
- apontador para a lista de vértices do modelo
- quantidade e próximo número disponível para regiões
- quantidade e próximo número disponível para cascas
- quantidade e próximo número disponível para faces
- quantidade e próximo número disponível para arestas
- quantidade e próximo número disponível para vértices
- quantidade e próximo número disponível para usoFaces
- quantidade e próximo número disponível para usoArestas
- quantidade e próximo número disponível para usoVértices
- apontador para lista de *labels* de geometria e malha do modelo

REGIÃO
- nome e código identificador da região
- apontador para a casca externa da região
- apontador para a lista de cascas da região
- apontador para o *label* com características físicas da região
- apontador para o modelo que contém a região (por intermédio do nó cabeça)

CASCA
- código identificador da casca
- apontador para a lista de usoFaces
- apontador para a lista de usoArestas
- apontador para a lista de usoVértices
- apontador para o *label* com características físicas da casca
- apontador para a região que contém esta casca (por intermédio do nó cabeça)

FACE
- código identificador da face
- apontadores para as duas ocorrências de usos da face
- apontador para a definição da geometria da face
- apontador para o *label* com características físicas da face

USOFACE
- código identificador da UsoFace
- apontador para o usoCiclo externo da usoFace
- apontador para a lista de usoCiclos da usoFace
- apontador para a face a qual este usoFace se refere
- orientação da normal da usoFace em relação à face (+ / -)
- apontador para a casca que contém a usoFace (por intermédio do nó cabeça)

USOCICLO
- apontador para a lista de semi-arestas do usoCiclo
- apontador para a usoFace que contém este usoCiclo (por intermédio do nó cabeça)

SEMI-ARESTA
- apontador para a usoAresta da qual ela faz parte
- apontador para o usoVértice do qual ela se origina
- apontador para o usoCiclo que contém a semiAresta (por intermédio do nó cabeça)

ARESTA
- código identificador da Aresta
- apontador para a lista auxiliar de usos da Aresta
- apontador para a definição da geometria da aresta
- apontador para o *label* com características físicas da aresta

USOARESTA
- código identificador da usoAresta
- apontadores para as duas semiArestas associadas
- apontador para o auxiliar que associa a outros usos da aresta
- apontador para a casca que contém a usoAresta (por intermédio do nó cabeça)

VÉRTICE
- código identificador do vértice
- apontador para a definição da geometria do vértice
- apontador para a lista auxiliar de usos do vértice
- apontador para o *label* com características físicas do vértice

USOVÉRTICE
- código identificador do usoVértice
- apontador o auxiliar que relaciona os usos do vértice
- apontador para a casca que contém o usoVértice (por intermédio do nó cabeça)

AUXILIAR DE USO DA ARESTA
- apontador para o usoAresta pertencente a uma lista de usoArestas de uma casca
- apontador para a aresta da qual ele é derivado (por intermédio do nó cabeça)

AUXILIAR DE USO DO VÉRTICE
- apontador para o usoVértice pertencente a uma lista de usoVértices de uma casca
- apontador para o vértice do qual ele é derivado (por intermédio do nó cabeça)

GEOM-FACE_
- plano da face (vetor normal e um ponto)
- limites mínimo e máximo do envelope convexo que contém a face

GEOM-ARESTA
- reta que contém a aresta (direção e um ponto)
- limites mínimo e máximo do envelope convexo que contém a aresta

GEOM-VÉRTICE
- coordenadas *x*, *y*, *z*
- graus de liberdade do vértice

Quadro V.5 – Descrição sucinta do conteúdo dos principais componentes

V.3.3 – OPERADORES DE EULER

O módulo *EulerOp* implementa os operadores de Euler, que garantem a construção de modelos B-rep topologicamente válidos. Apresenta-se a seguir uma descrição detalhada dos operadores desenvolvidos, incluindo a relação e o modo de identificação de seus parâmetros. As operações serão executadas sobre o modelo atual, controlado pelo estado geral do sistema. Ele está orientado de maneira que, seguindo o ciclo de semi-arestas contido nos usoFaces e utilizando a regra da mão direita, a normal à face aponta para o exterior do sólido que está sendo criado [Mag94a].

Vale ressaltar mais uma vez que os operadores de Euler deverão ser aplicados para gerar os elementos face, aresta e vértice bem como o uso desses elementos, estando as relações topológicas definidas sobre o uso, e não sobre os elementos propriamente ditos, porém, para simplicidade do texto, a explicação dos operadores não faz distinção entre o elemento e seu uso.

V.3.3.1 – MVFR E KVFR

O operador MVFR ("*Make Vertex Face Region*") cria a partir do nada uma instância da estrutura de dados de um sólido que possui um único vértice, uma face e uma casca externa (região). A nova face possui um ciclo vazio, sem nenhuma aresta. O "sólido" criado não satisfaz a noção intuitiva de sólido, constituindo uma "forma esquelética", porém é útil como estágio inicial na criação de sólidos B-rep por meio de uma seqüência de operadores de Euler. KVFR ("*Kill Vertex Face Region*") constitui a operação oposta, que destrói a instância esquelética criada com uma estrutura de dados igual à do MVFR. A representação gráfica destes dois operadores é apresentada na figura V.16a e a estrutura de dados gerada ao ser aplicado o MVFR, na figura V.16b. Os parâmetros utilizados por esses operadores estão descritos no quadro V.6.

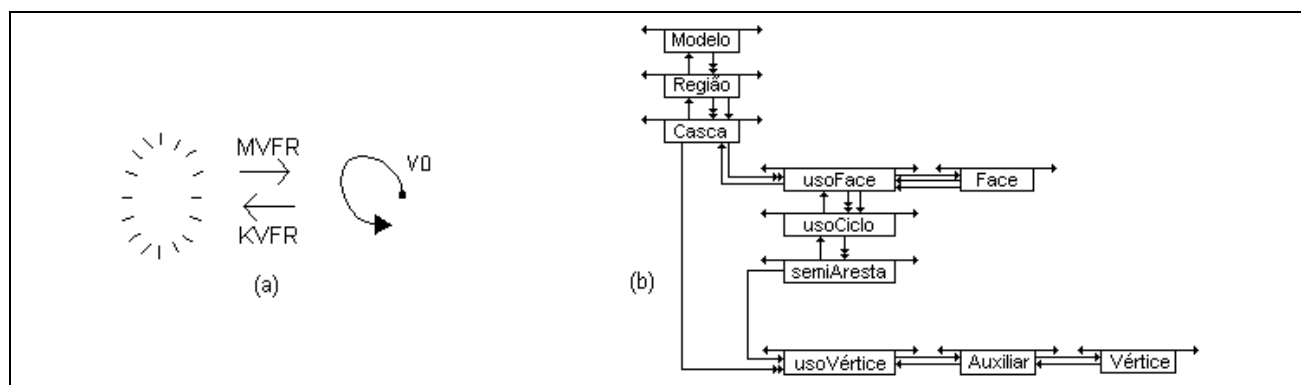



Figura V.16 – Efeito obtido ao serem aplicados os operadores MVFR e KVFR

Operação	Parâmetros
MVFR	vn: identificador do vértice a ser criado coord: ponto contendo as coordenadas x, y, z do vértice a ser criado gl: graus de liberdade do vértice criado – fixo (NONE_DF) ou pode movimentar-se sobre a aresta (ONE_DF), o plano (TWO_DF) ou o volume(THREE_DF) obs - o nome da região bem como os identificadores de região, casca e face possuem geração automática
KVFR	rn: identificador da região a ser destruída, que deve estar na forma esquelética

Quadro V.6 – Parâmetros utilizados nas operações MVFR e KVFR

V.3.3.2 – MEV E KEV

Os operadores MEV ("*Make Edge Vertex*") e KEV ("*Kill Edge Vertex*") manipulam propriedades topológicas locais de um modelo por fronteira, ou seja, alteram a topologia do modelo sem interferir no número total de cascas e cavidades. MEV subdivide um vértice em dois, concatenando-os por uma nova aresta, tendo como efeito a adição de um vértice e uma aresta à estrutura de dados. Podem ser utilizados em três situações distintas, ilustradas na figura V.17:

(a) partindo da forma esquelética, na qual o ciclo existente é vazio – o vértice sem aresta é dividido em dois, que são unidos pela aresta criada, que por sua vez contém duas semi-arestas com direções opostas. Ainda não faz sentido falar em vértice anterior e posterior;

(b) gerando uma aresta pertencente a uma única face – neste caso, o vértice criado será posicionado antes da semi-aresta fornecida como parâmetro, considerando a orientação do ciclo de semi-arestas;

(c) gerando uma aresta pertencente a duas faces – as semi-arestas a serem fornecidas como parâmetro são as que saem do vértice a ser dividido, e o vértice criado ficará antes destas semi-arestas na seqüência do ciclo.

O operador KEV desfaz qualquer um destes casos. Dada uma aresta conectando dois vértices distintos, KEV remove a aresta e une os dois vértices, deixando apenas um. Os parâmetros utilizados pelos operadores MEV e KEV são apresentados no quadro V.7.

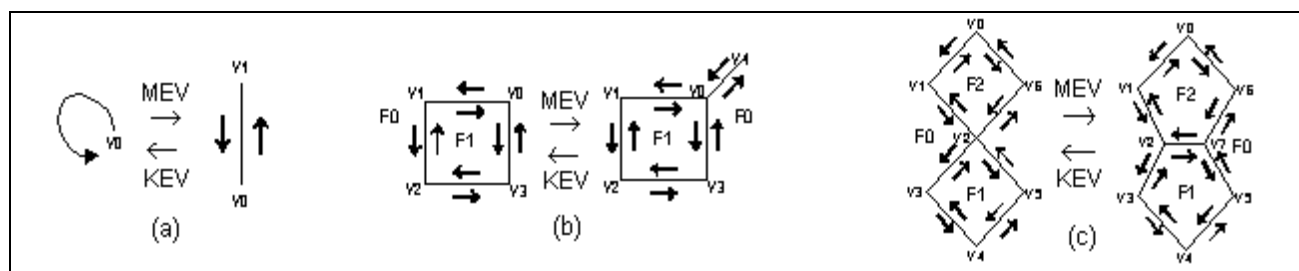



Figura V.17 – Formas de utilização das operações MEV e KEV

OPERAÇÃO	PARÂMETROS
MEV	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da primeira face onde o vértice e a aresta serão criados fn2: identificador da segunda face onde o vértice e a aresta serão criados vn1: identificador do vértice onde saem as 2 semi-arestas de referência vn2: identificador do vértice onde chega a primeira semi-aresta (em fn1) vn3: identificador do vértice onde chega a segunda semi-aresta (em fn2) vn4: identificador do vértice a ser criado e unido a vn1 pela nova aresta coord: ponto contendo as coordenadas x, y, z do vértice a ser criado gl: graus de liberdade do vértice criado – fixo (NONE_DF) ou pode movimentar-se sobre a aresta (ONE_DF), o plano (TWO_DF) ou o volume(THREE_DF)
KEV	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face que terá a aresta e o vértice destruídos vn1: identificador do vértice a ser destruído, de onde sai a semi-aresta vn2: identificador do vértice onde chega a semi-aresta a ser destruída, ou seja, o identificador do vértice que será unido (em fn1)

Quadro V.7 – Parâmetros utilizados nas operações MEV e KEV

V.3.3.3 – MEF E KEF

Da mesma forma que MEV e KEV, MEF ("*Make Edge Face*") e KEF ("*Kill Edge Face*") manipulam propriedades topológicas locais. MEF subdivide um ciclo, fazendo a ligação de dois vértices por uma nova aresta, ou seja, adiciona uma nova aresta e uma nova face à estrutura de dados. Cuidado especial deve ser tomado ao identificar as semi-arestas utilizadas no fechamento da nova face, pois elas interferem na orientação da face gerada. A nova aresta deve ser especificada pelas semi-arestas que saem de seus dois vértices, o inicial e o final, seguindo o ciclo da nova face. Estas operações podem ocorrer em quatro situações distintas, ilustradas na figura V.18:

(a) criando uma aresta que define a primeira face delimitada do sólido – como mencionado, a orientação desta aresta influenciará na orientação de todo o sólido. A aresta a ser criada é especificada pelas duas semi-arestas que saem dos vértices, ligados pela nova aresta. A face criada é a que foi fechada;

(b) criando uma aresta que divide uma face delimitada existente – os vértices a serem ligados pela aresta devem ser especificados considerando as semi-arestas que partem deles, seguindo o sentido do ciclo da nova face;

(c) criando aresta e face a partir do modelo esquelético – em analogia ao ocorrido com MEV, MEF pode ser aplicado a ciclos vazios, resultando em um único vértice com uma aresta circular separando duas faces;

(d) criando aresta e face nulas – neste caso, a aresta é gerada com os vértices inicial e final coincidentes, e a orientação da nova face é oposta à da face de onde foram derivadas.

A operação KEF desfaz o que a MEF fez. Dada uma aresta adjacente a duas faces distintas, KEF remove a aresta, une as duas faces em uma única, e mescla os ciclos externos das faces envolvidas. Os parâmetros utilizados em ambas operações são descritos no quadro V.8.

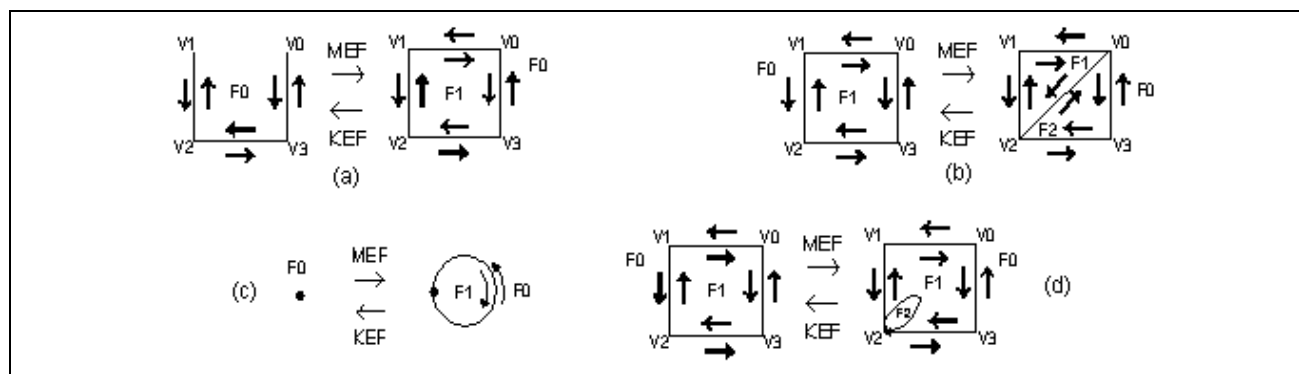



Figura V.18 – Formas de utilização das operações MEF e KEF

OPERAÇÃO	PARÂMETROS
MEF	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face onde a aresta e a nova face serão criadas vn1: identificador do vértice de onde sai a semi-aresta a ser criada vn2: identificador do vértice onde chega a semi-aresta, já existente, que sai de vn1 (em fn1) vn3: identificador do vértice onde chega a semi-aresta a ser criada vn4: identificador do vértice onde chega a semi-aresta, já existente, que sai de vn3 (em fn1) obs - o identificador da face criada possui geração automática
KEF	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face a ser destruída, que contém a aresta vn1: identificador do vértice de onde sai a semi-aresta a ser destruída vn2: identificador do vértice onde chega a semi-aresta a destruir (em fn1)

Quadro V.8 – Parâmetros utilizados nas operações MEF e KEF

V.3.3.4 – KEML E MEKL

KEML e MEKL constituem operadores de conveniência, cuja necessidade é devida mais à convenção utilizada no esquema de representação do que à teoria. Para fazer uso completo de ciclos vazios, é necessário um operador específico para a sua criação. KEML ("*Kill Edge Make Loop*") divide um ciclo em dois novos, removendo a aresta que nele aparece duas vezes. Separa ciclos conectados, que se tornam ciclos independentes, gerando anéis – seqüência de vértices que possuem um ciclo de semi-arestas somente em uma das direções. Possui ainda o efeito de remover uma aresta e adicionar um ciclo à estrutura de dados.

O operador inverso MEKL ("*Make Edge Kill Loop*") une dois ciclos de uma face, pela inclusão de uma aresta ligando um vértice de cada ciclo. O caso geral é ilustrado na figura V.19a e os casos especiais, nos quais um ou ambos os ciclos resultantes são nulos, correspondem respectivamente às figuras V.19b e V.19c. Os parâmetros utilizados por estes operadores estão descritos no quadro V.9. A semi-aresta "*vn1-vn2*" utilizada pela operação KEML deve ser escolhida de tal forma que as semi-arestas seguintes, até a "*vn2-vn1*" pertençam ao novo ciclo, e as subseqüentes pertençam ao ciclo antigo (externo).

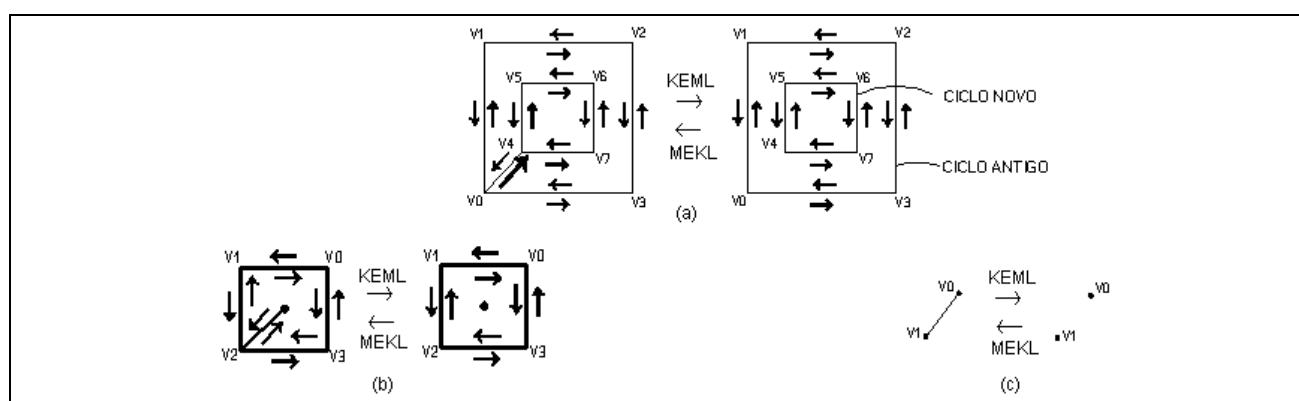



Figura V.19 – Formas de utilização das operações KEML e MEKL

OPERAÇÃO	PARÂMETROS
KEML	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face que terá a aresta destruída e o ciclo criado vn1: identificador do vértice inicial da aresta a ser destruída vn2: identificador do vértice final da aresta a ser destruída
MEKL	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face que terá o ciclo destruído e a aresta criada vn1: identificador do vértice a ser unido, contido no ciclo externo da face vn2: identificador do vértice onde chega a semi-aresta, já existente, que sai de vn1 (em fn1) vn3: identificador do vértice a ser unido, contido no ciclo da face vn4: identificador do vértice onde chega a semi-aresta, já existente, que sai de vn3 (em fn1)

Quadro V.9 – Parâmetros utilizados nas operações KEML e MEKL

V.3.3.5 – MFKLH E KFMLH

Os operadores MFKLH e KFMLH manipulam propriedades topológicas globais. KFMLH ("*Kill Face Make Loop Hole*") realiza a operação de soma conectada, definida para modelos planares: dadas duas faces F1 e F2, este operador une-as em uma única face, transformando o ciclo de fronteira F2 em um ciclo de F1. Seu efeito então é o de remover uma face (F2) e adicionar um ciclo. É difícil ilustrar essa operação, uma vez que não possui efeito no arranjo local de arestas e vértices,

constituindo uma manipulação global real. KFMLH é uma denominação imprópria, pois o operador só criará o buraco ("*hole*") se as duas faces envolvidas pertencerem à mesma casca. Quando aplicado a faces pertencentes a cascas distintas, seu efeito é combiná-las em uma única casca, e o nome KFSML ("*Kill Face Shell Make Loop*") seria mais apropriado, apesar de ainda não ser o ideal.

MFKLH ("*Make Face Kill Loop Hole*") é o operador inverso, que modifica o ciclo de uma face para um ciclo na fronteira da nova face. O item (a) da figura V.20 mostra a representação gráfica destes operadores, enquanto o item (b) ilustra o ocorrido com o sólido. O quadro V.10 descreve os parâmetros envolvidos nestas operações.

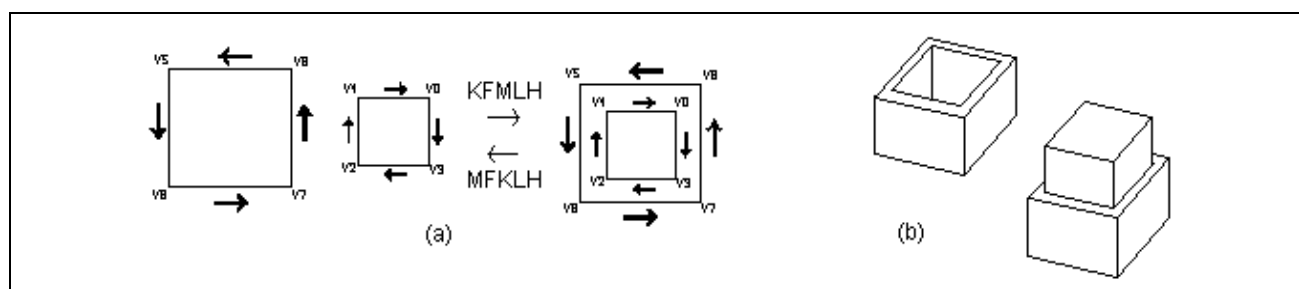



Figura V.20 – Efeito obtido ao serem aplicados os operadores KFMLH e MFKLH

OPERAÇÃO	PARÂMETROS
MFKLH	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face contendo o ciclo e o buraco a serem destruídos vn1: identificador do vértice inicial de uma semi-aresta contida no ciclo vn2: identificador do vértice final da semi-aresta definida em vn1 obs - o identificador da face criada possui geração automática
KFMLH	rn: identificador da região sobre a qual ocorrerá a operação cn: identificador da casca sobre a qual ocorrerá a operação fn1: identificador da face que conterá o ciclo e o buraco fn2: identificador da face que será destruída

Quadro V.10 – Parâmetros utilizados nas operações MFKLH e KFMLH

V.3.3.6 – MSKR E KSMR

O operador MSKR ("*Make Shell Kill Region*") transfere a casca externa de um sólido para outro e, em seguida, elimina o sólido que continha a casca. A casca transferida passa a ser uma casca interna do sólido que a recebeu. Compondo sua fronteira, o sólido a ser eliminado deve possuir apenas a casca a ser transferida. O operador inverso KSMR ("*Kill Shell Make Region*") realiza a operação inversa, criando um sólido cuja fronteira é composta por uma casca interna transferida de outro sólido. A figura V.21 mostra graficamente como estas operações ocorrem, e o quadro V.11 descreve os parâmetros envolvidos nestas operações.

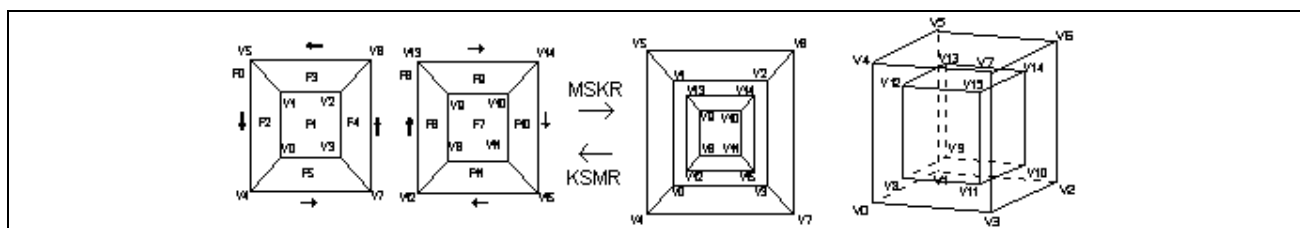



Figura V.21 – Representação gráfica das operações MSKR e KSMR

OPERAÇÃO	PARÂMETROS
MSKR	rn1: identificador da região que receberá a nova casca rn2: identificador da região que será destruída
KSMR	rn1: identificador da região que sofrerá a perda da casca cn1: identificador da casca de rn1 sobre a qual ocorrerá a operação (deverá ser uma casca interna) obs - o nome e o código identificador da região possuem geração automática

Quadro V.11 – Parâmetros utilizados nas operações MSKR e KSMR

V.3.3.7 – MFFR E KFFR

O operador MFFR ("*Make double Face Region*") divide uma região em duas, que permanecem com uma fronteira comum, definida por dois usos de uma mesma face. Os vértices e arestas desta face de referência já devem estar criados. O operador inverso KFFR ("*Kill double Face Region*") realiza a operação contrária, destruindo a face compartilhada entre duas regiões e seus usos correspondentes. Desta forma, uma das regiões também é destruída, permanecendo uma única região englobando todo o volume anteriormente dividido. A figura V.22 mostra graficamente como estas operações ocorrem, e o quadro V.12 descreve os parâmetros envolvidos nestas operações.

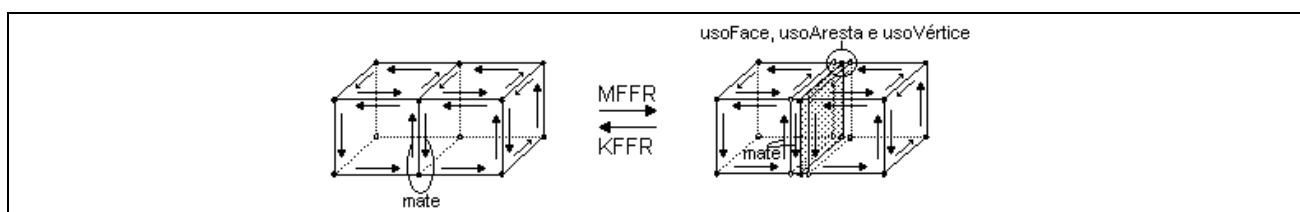



Figura V.22 – Representação gráfica das operações MFFR e KFFR

OPERAÇÃO	PARÂMETROS
MFFR	rn1: identificador da região que sofrerá a divisão cn1: identificador da casca que sofrerá a divisão fn1: identificador da face que contém a semi-aresta que vai de vn1 a vn2 fn2: identificador da face que contém a semi-aresta que vai de vn2 a vn3 vn1: identificador do vértice inicial de uma das semi-arestas que definirá o plano de corte vn2: identificador do vértice final da semi-aresta de origem em vn1 e inicial de uma outra semi-aresta que definirá o plano de corte vn3: identificador do vértice final da semi-aresta de origem em vn2 obs - o nome e o código identificador da região possuem geração automática
KFFR	rn1: identificador da região que englobará a outra região rn2: identificador da região que será destruída fn: identificador da face que será destruída e causará a destruição de seus usos de face

Quadro V.12 – Parâmetros utilizados nas operações MFFR e KFFR

V.3.3.8 – OPERADORES ADICIONAIS GR, UR, GF E UF

O operador GR ("*Glue Regions"*) une duas regiões pela colagem de duas faces idênticas e coincidentes, cada uma pertencente a uma das regiões. Primeiro, a estrutura de dados de uma das regiões recebe os dados da estrutura da outra região e gera uma única estrutura representando a união. As faces idênticas são então eliminadas, bem como as arestas e vértices de uma das faces.

A operação inversa UR ("*Unglue Region*") realiza o processo oposto, duplicando as arestas e os vértices necessários, criando duas faces independentes e separando as duas regiões. A figura V.23a ilustra esta operação e a figura V.23b mostra a seqüência de operações aplicadas para a junção das faces e geração do “buraco”.

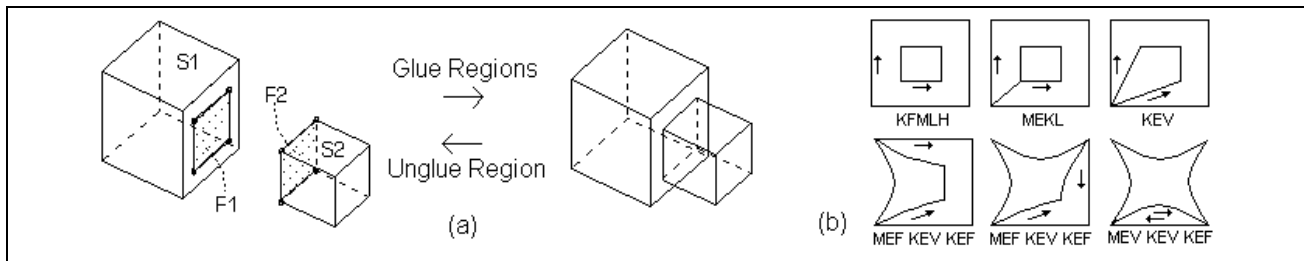



Figura V.23 – Efeito obtido ao serem aplicados os operadores *Glue Regions* e seu inverso, *Unglue Region*

Quando duas regiões devem ser unidas pela colagem de mais de duas faces idênticas, o operador GF ("*Glue Faces*") é usado após o operador *Glue Regions* para unir as demais faces idênticas das regiões não coladas por *Glue Regions*. Estas faces são então removidas, juntamente com os vértices e arestas de uma das faces, da mesma forma que ocorre em *Glue Regions.* A operação UF ("*Unglue Face*") realiza o processo inverso, duplicando as arestas e os vértices necessários e desmembrando a face em duas independentes. Pode-se notar que os operadores *Glue Regions* e *Glue Faces* fazem exatamente a mesma coisa, diferindo-se apenas pela mesclagem das regiões – atribuição de todas as cascas da região que vai ser destruída para a que vai contê-la. A figura V.24 elucida este caso, e o quadro V.13 descreve os parâmetros envolvidos nestas quatro operações.

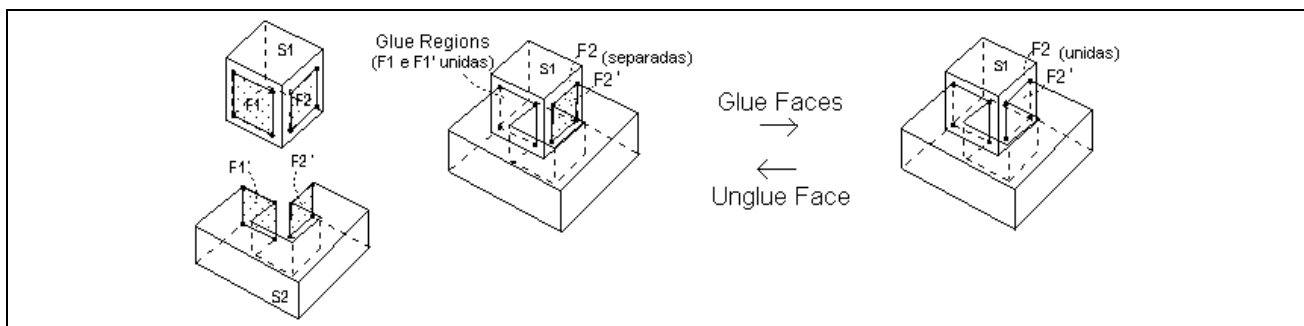



Figura V.24 – Efeito obtido ao serem aplicados os operadores *Glue Faces* e seu inverso, *Unglue Face*

OPERAÇÃO	PARÂMETROS
Glue Regions	rn1: identificador da região sobre a qual ocorrerá a operação rn2: identificador da região a ser colada em rn1 fn1: identificador da face da região rn1 que será colada fn2: identificador da face da região rn2 que será colada
Unglue Region	rn: identificador da região que sofrerá o desmembramento cn: identificador da casca sobre a qual ocorrerá o desmembramento fn1: identificador da face que contém a semi-aresta que vai de vn1 a vn2 fn2: identificador da face que contém a semi-aresta que vai de vn2 a vn3 vn1: identificador do vértice inicial de uma das semi-arestas que definirá o plano de corte vn2: identificador do vértice final da semi-aresta de origem em vn1 e inicial de uma outra semi-aresta que definirá o plano de corte vn3: identificador do vértice final da semi-aresta de origem em vn2 obs - o nome e o código identificador da região possuem geração automática; os códigos identificadores das faces a serem criadas também possuem geração automática
Glue Faces	rn1: identificador da região sobre a qual ocorrerá a operação fn1: identificador de uma das faces da região que será colada fn2: identificador da outra face da região que será colada
Unglue Face	rn: identificador da região que sofrerá o desmembramento da face cn: identificador da casca que sofrerá o desmembramento da face fn1: identificador da face que contém a semi-aresta que vai de vn1 a vn2 fn2: identificador da face que contém a semi-aresta que vai de vn2 a vn3 vn1: identificador do vértice inicial de uma das semi-arestas que definirá o plano de corte vn2: identificador do vértice final da semi-aresta de origem em vn1 e inicial de uma outra semi-aresta que definirá o plano de corte vn3: identificador do vértice final da semi-aresta de origem em vn2 obs - os códigos identificadores das faces a serem criadas possuem geração automática

Quadro V.13 – Parâmetros utilizados nas operações GR, UR, GF e UF

V.3.3.9 – OPERADORES ADICIONAIS AF, DF, AR E DR

O operador AF ("*Assemble Faces"*) cria uma fronteira única entre duas regiões a partir de duas faces idênticas e coincidentes, cada uma pertencente a uma região. Este operador se distingue do *Glue Faces* por não destruir a fronteira existente, mas reconstruí-la de forma única.. As faces idênticas são eliminadas e uma face de fronteira é gerada, contendo as ocorrências de uso da face em cada uma das regiões envolvidas. Este mesmo procedimento se repete para ciclos, arestas e vértices.

A operação inversa DF ("*DisassembleFace*") realiza o processo oposto, duplicando as arestas e os vértices necessários, criando duas faces independentes e separando as duas regiões. A figura V.25a ilustra estas operações, e o quadro V.14 descreve os principais parâmetros envolvidos.

O operador AR ("*Assemble Regions"*) também cria uma fronteira única entre duas regiões, mas para todas as faces idênticas e coincidentes que ocorrem em ambas as regiões. Além de não destruir a fronteira existente, este operador se distingue do *Glue Regions* por não mesclar as estruturas de dados das regiões. Ele pode ser visto como aplicações sucessivas do *Assemble Faces* sobre todas as faces coincidentes entre duas regiões. Da mesma forma, a operação inversa DR ("*Disassemble Regions*") duplica arestas e vértices e cria faces independentes, separando as duas regiões. A figura V.25b exemplifica estas operações.

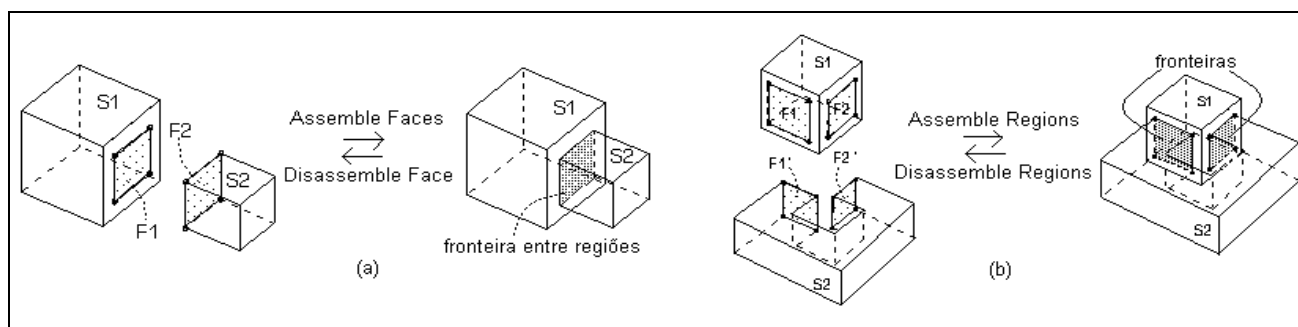



Figura V.25 – Efeito obtido ao serem aplicados os operadores *Assemble Faces*, *Assemble Regions* e seus inversos

OPERAÇÃO	PARÂMETROS
Assemble Faces	rn1: identificador da região 1 sobre a qual ocorrerá a operação rn2: identificador da região 2 sobre a qual ocorrerá a operação sn1: identificador da casca da região 1 sobre a qual ocorrerá a operação sn2: identificador da casca da região 2 sobre a qual ocorrerá a operação fn1: identificador da face da região rn1 que será montada fn2: identificador da face da região rn2 que será montada
Disassemble Face	rn: identificador da região sobre a qual ocorrerá a operação sn: identificador da casca sobre a qual ocorrerá a operação fn: identificador da face de fronteira que será desmontada obs - o código identificador da face a ser criada possui geração automática
Assemble Regions	rn1: identificador da região 1 sobre a qual ocorrerá a operação rn2: identificador da região 2 sobre a qual ocorrerá a operação sn1: identificador da casca da região 1 sobre a qual ocorrerá a operação sn2: identificador da casca da região 2 sobre a qual ocorrerá a operação
Disassemble Regions	rn1: identificador da região 1 sobre a qual ocorrerá a operação rn2: identificador da região 2 sobre a qual ocorrerá a operação sn1: identificador da casca da região 1 sobre a qual ocorrerá a operação sn2: identificador da casca da região 2 sobre a qual ocorrerá a operação obs - o nome e o código identificador da região possuem geração automática; os códigos identificadores das faces a serem criadas também possuem geração automática

Quadro V.14 – Parâmetros utilizados nas operações de montagem AF e DF

V.3.4 – A AVALIAÇÃO DA FRONTEIRA

Baseado principalmente no trabalho de Muuss e Butler [Muu91], e com aplicação de idéias apresentadas por Pilz e Kamel [Pil89], Gursoz et al.[Gur91] e Mäntylä [Män83, Män86, Män88], o processo de avaliação da fronteira desenvolvido para este modelador segue a abordagem incremental: a partir da fronteira definida sobre as primitivas, avalia-se o modelo para cada operação boolena presente na árvore CSG, gerando a fronteira da sub-árvore correspondente. A representação B-rep resultante do modelo é obtida após a avaliação do operador que está presente na raiz da árvore CSG. A avaliação da fronteira é acompanhada da aproximação poligonal, e os passos de cada etapa são mostrados no fluxograma da figura V.26. Como resultado de cada etapa, é gerada uma malha poligonal consistente e não redundante, que pode ser usada em uma etapa posterior da

avaliação, até a obtenção da fronteira completa do modelo. Uma base de dados consistente, que forneça meios para armazenar, acessar e referenciar a informação geométrica e topológica dos objetos sólidos, é fundamental para o processo.

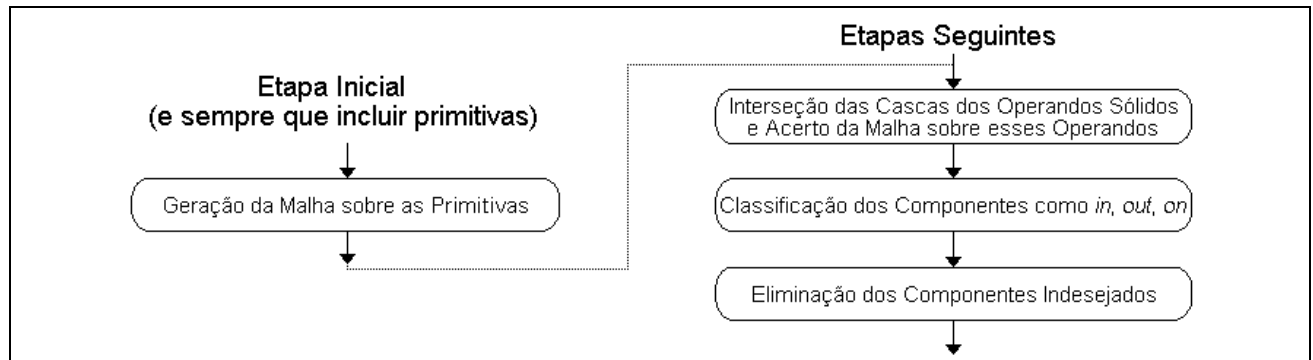



Figura V.26 – Fluxograma para as etapas da avaliação incremental da fronteira

V.3.4.1 – GERAÇÃO DA MALHA SOBRE AS PRIMITIVAS

O primeiro passo na avaliação da fronteira processa um operando sólido por vez, gerando sobre ele uma malha de polígonos estruturada e homogênea, que garante a inexistência de interseção entre eles. Esta etapa deverá ocorrer somente sobre primitivas instanciadas ou definidas por varredura, pois os operandos que correspondem a sub-árvores CSG já tiveram sua fronteira avaliada em alguma etapa anterior deste processo incremental. Naturalmente, cada tipo de primitiva requer uma função de geração de malha específica, porém, em geral, este modelador gera uma malha triangularizada sobre as primitivas definidas, ou seja, todas as faces geradas na superfície das primitivas possuem forma triangular.

Na conversão de sólidos primitivos curvos para a representação por facetas poligonais, existem dois aspectos a serem considerados: o estabelecimento da topologia adotada na aproximação e a geração da geometria associada a essa topologia. Como o processo de divisão em facetas está diretamente relacionado à geração da malha sobre as primitivas, tais aspectos serão detalhados no próximo capítulo, que descreve o Subsistema de Geração de Malha. A conversão de superfícies curvas em facetas planares será certamente inexata[Seg90]. Para fornecer um referencial sobre a natureza e magnitude dos erros introduzidos, três controles de tolerância podem ser considerados [Muu91]:

— a *tolerância absoluta*, que limita a diferença máxima permitida entre qualquer ponto na aproximação e seu correspondente no sólido original. Permite assegurar que nenhuma face desvie-se da superfície correta mais do que seu valor, expresso como a distância absoluta na unidade utilizada;

— A *tolerância relativa*, que também limita o erro máximo em qualquer ponto, mas é expressa como uma fração entre 0.0 e 1.0 do diâmetro da esfera limítrofe que envolve o sólido original. Permite assegurar que nenhuma face desvie-se da superfície real mais do que um percentual em relação ao tamanho do sólido;

— A *tolerância normal*, que limita o erro angular máximo da normal à superfície. Permite assegurar que a normal à superfície de todas as faces aproximadas não se desvie, em relação às normais à superfície exata nos pontos correspondentes do sólido, mais que o valor angular definido.

Essas tolerâncias podem ser usadas sozinhas ou em qualquer combinação: se mais de uma tolerância for especificada, a mais restritiva deverá ser aplicada. Optou-se por incluir no modelador a tolerância absoluta, à qual é atribuído um valor mínimo padrão, caso seu valor não seja fornecido.

V.3.4.2 – INTERSEÇÃO DAS CASCAS DOS OPERANDOS SÓLIDOS

Até esse estágio, a interferência entre as malhas dos operandos foi ignorada. O cálculo das interseções entre as malhas é a parte central e mais importante da avaliação da fronteira. A estratégia utilizada para interseção e corte das duas cascas, uma em relação à outra, consiste em varrer todas as entidades dos operandos para detectar quais as que possuem interseção no processo. A fim de evitar computação cara e desnecessária, a interferência espacial global entre os dois operandos pode ser examinada antes de suas interseções locais, por meio do teste dos envelopes convexos correspondentes. O processo é resumido pelo algoritmo descrito no quadro V.15.

```
se  "os envelopes convexos das cascas A e B se sobrepõem"
   então  para  "cada face da casca A" faça
             se  "o envelope convexo da face A sobrepõe-se ao envelope convexo da casca B"
                então  para  "cada face na casca B" faça
                          se  "os envelopes convexos das faces A e B se sobrepõem"
                             então  "intercepte as arestas da face A com o plano da face B"
                                    "intercepte as arestas da face B com o plano da face A"
                                    "insira nova topologia e realize o malhamento"
```

Quadro V.15 – Processo para cálculo da interseção entre as cascas

O processo de comparar os envelopes convexos da face consiste em verificar não só se estes se sobrepõem de alguma forma, mas também se ocorre a sobreposição ao longo da linha de interseção, o que é importante para reduzir o número de interseções de faces realizadas.

Quando os envelopes convexos de duas faces se sobrepõem, significa que elas se interceptam. As equações do plano dessas faces são então comparadas entre si quanto à igualdade: se as duas faces forem coplanares, pode ser usada uma abordagem de recorte de polígono bidimensional – no caso deste modelador, específica para faces triangulares – a fim de obter a interseção de seus ciclos. Caso as duas faces não sejam coplanares, a interseção deve ser calculada de forma a obter o vértice ou a aresta que estão sendo compartilhados. Uma lista contendo todos os vértices que estão na linha de interseção entre os dois planos deve ser gerada, tendo por base o cálculo da interseção de cada aresta da face A com o plano da face B e de cada aresta da face B com o plano da face A. O processo de interseção realiza-se da seguinte forma [Muu91]: se um ponto extremo de uma aresta encontra-se topológica e geometricamente no plano da outra face, aquele vértice é registrado na lista de pontos pertencentes à linha de interseção; se a aresta atravessa o plano da outra face, essa aresta é dividida em duas, e o vértice gerado também está sobre a linha de interseção, devendo ser adicionado à lista de pontos.

Após a determinação de todas as interseções, essa lista de pontos é classificada geometricamente ao longo da linha de interseção e usada para determinar quais segmentos são realmente compartilhados entre as duas faces, devendo ser a elas adicionados. Estes segmentos pertencem a uma das três categorias:

– os dois pontos estão no contorno da face: após certificar-se de que a aresta ainda não existe, os dois pontos extremos devem ser examinados e tratados conforme ilustrado na figura V.27a: se eles são parte do mesmo ciclo da face, este ciclo é dividido em dois ciclos separados, que compartilham a nova aresta; se eles pertencem a ciclos diferentes, a aresta que conecta os dois ciclos é criada, e eles são unidos em um ciclo único. Para modelos triangularizados, ciclos internos não existem, e as faces resultantes deverão ser novamente triangularizadas considerando os pontos inseridos no contorno, os quais podem estar em arestas distintas, na mesma aresta ou sobre algum vértice;

– um ponto está no contorno da face: o ciclo que contém o vértice existente é estendido para incluir o novo ponto, conforme ilustrado na figura V.27b. Para modelos triangularizados, a inserção do ponto em uma aresta do contorno leva a uma nova triangularização da face, mesmo que a interseção resulte somente nesse ponto, uma vez que é necessário manter a compatibilidade da malha superficial dos operandos;

– nenhum ponto está no contorno da face: um novo ciclo interior deve ser criado na face, como mostra a figura V.27c. Para modelos triangularizados, esses pontos devem sempre ser unidos aos vértices da face inicial, de forma a gerar novos triângulos.

Quando toda a topologia apropriada houver sido inserida nas faces, a topologia das duas faces deve estar conectada de tal forma que as arestas e vértices que possam ser compartilhados entre as duas faces estejam realmente sendo compartilhados. Este processo consiste principalmente na atribuição de usoVértices a vértices comuns, além da atribuição orientada dos usoArestas compartilhados por cada aresta.

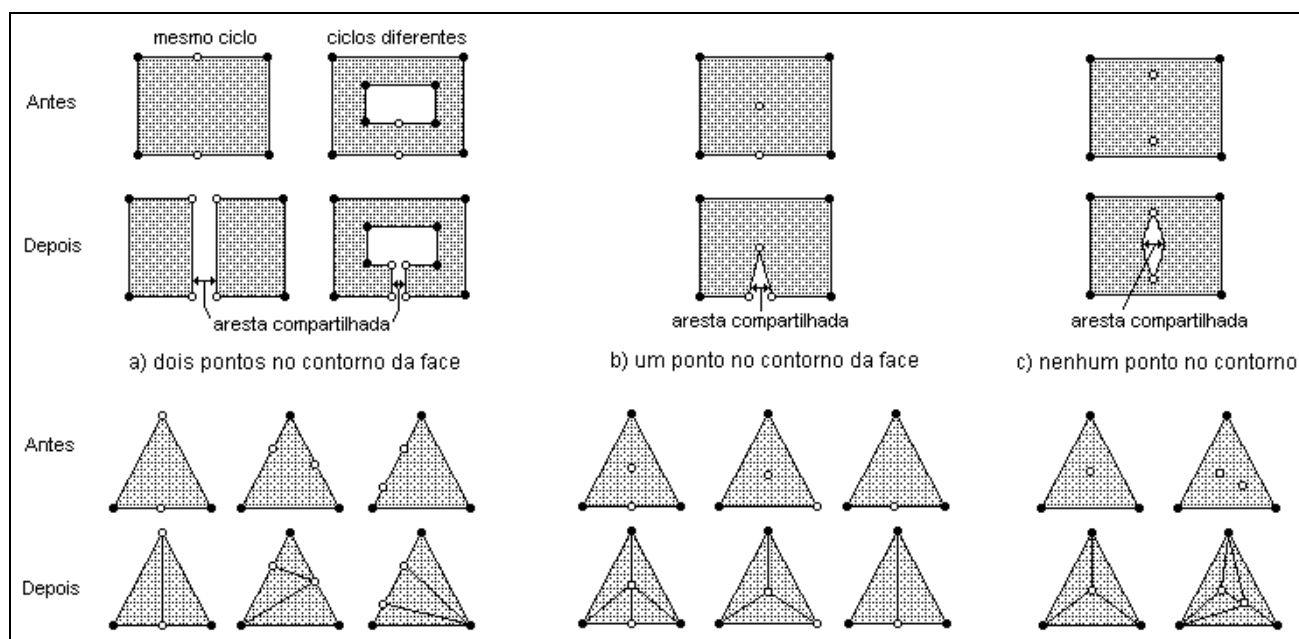



Figura V.27 – Interseção de faces não coplanares: inserção de segmentos em modelos poligonais e triangularizados

A abordagem para verificação da interseção entre faces triangulares coplanares segue a idéia proposta por Sutherland-Hodgman para recorte de quaisquer polígonos [Fol90]. Cada aresta de um triângulo é testada em relação às arestas do outro, podendo levar à divisão das arestas do contorno e ao acréscimo de novas arestas. Vários polígonos podem resultar da interseção dos dois triângulos, como mostra a figura V.28, sendo necessário proceder a uma abordagem especialmente organizada para tratar todos os casos possíveis e ainda triangularizá-los.

V.3.4.3 – Classificação dos Componentes

Após a realização do cálculo de todas as interseções, todas as entidades de uma casca deverão ser classificadas com relação à outra casca: cada face, ciclo, aresta e vértice deve ser

classificado como estando dentro (*in*), na superfície (*on*) ou fora da outra casca (*out*). Este processo é facilitado pelas informações topológicas disponíveis, bem como pela classificação de toda uma casca, por vez, em relação à outra casca.

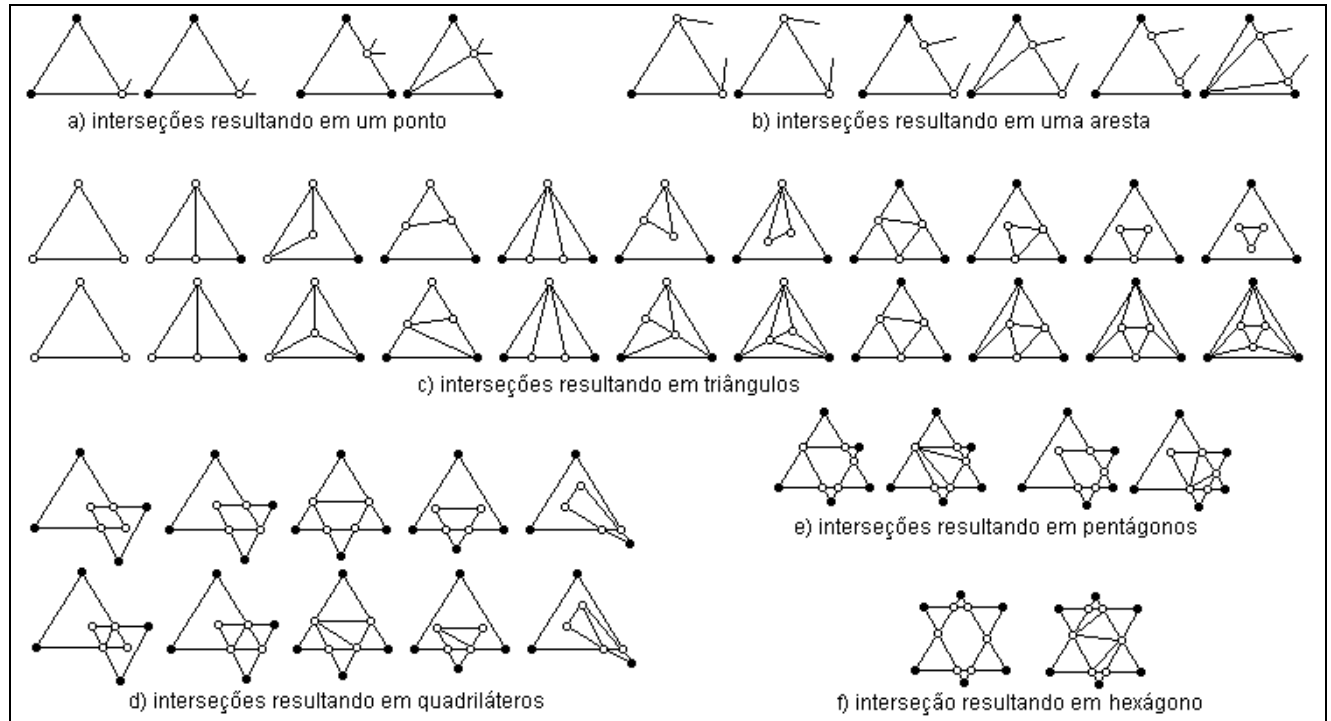



Figura V.28 – Casos possíveis na interseção de dois triângulos e inserção de segmentos de forma a manter a triangulação

A primeira classificação a ser estabelecida é a do vértice. Obviamente, se um vértice já tiver sido classificado em relação a uma casca B, qualquer uso daquele vértice presente na casca A compartilha da mesma classificação. Se não confirmada essa hipótese, será necessário proceder à classificação. Inicialmente, deve-se verificar se existe um outro uso do vértice sobre a topologia da casca B, o que o classificaria como *on*. Caso isso não ocorra, a geometria deve ser usada para determinar se o vértice está dentro ou fora da casca B. A verificação de que o vértice está fora do envelope convexo já o classificaria como *out*. Se esse recurso falhar, deve ser empregada a técnica de *raytracing*, ilustrada na figura V.29, que permite determinar a posição do vértice pela análise das interseções geradas entre um raio disparado arbitrariamente pelo vértice e as faces que compõem a casca: se o número de interseções geradas for ímpar, o vértice está dentro; se esse número for par, está fora.

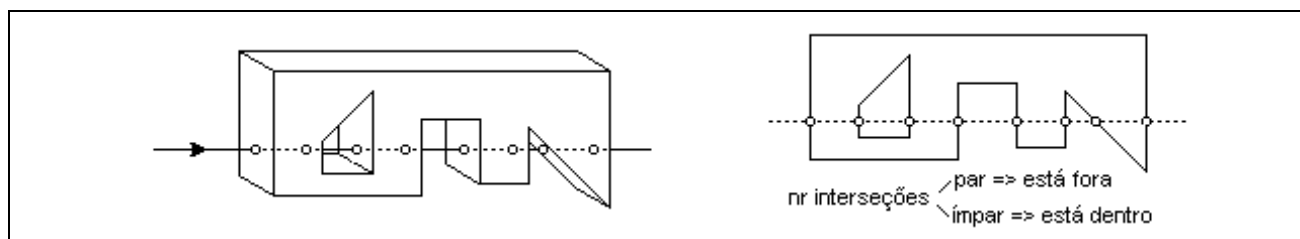



Figura V.29 – Utilização do *raytrace* para a classificação de vértices

Uma vez classificados todos os vértices da casca A em relação à casca B, é possível classificar as arestas de acordo com a classificação dos pontos extremos: se um ou ambos os extremos não estão "sobre" a casca B, a aresta obtém sua classificação do vértice; se os dois extremos estão "sobre" a casca B, o ponto médio da aresta é computado e classificado seguindo a técnica de *raytracing*, como descrito anteriormente. Uma aresta com extremos dentro e fora da casca B indica erro no processo de interseção. O quadro V.16 resume essa classificação.

pontos extremos	*out* / *out*	*out* / *on*	*on* / *on*	*in* / *on*	*in* / *in*	*in* / *out*
aresta	*out*	*out*	*raytracing*	*in*	*in*	erro

Quadro V.16 – Classificação das arestas por seus pontos extremos [Muu91]

A classificação dos ciclos ocorre da seguinte maneira: um ciclo composto por um único vértice herda a classificação do vértice; um ciclo de arestas que contém uma aresta que não está "sobre" a casca B herda essa classificação; um ciclo com arestas tanto dentro quanto fora da casca B indica erro no processo de interseção. Outras duas condições são requeridas para que um ciclo seja classificado como estando "sobre" a casca B: primeiro, deve existir um ciclo na topologia da casca B que possua exatamente o mesmo conjunto de arestas; segundo, o ciclo deve ser classificado como sendo *shared* – possui um correspondente na outra casca, com as normais apontando na mesma direção – ou *anti-shared* – idem, com as normais à superfície apontando em direções opostas –, como mostrado na figura V.30 . Finalmente, as faces são classificadas segundo seus ciclos externos. As combinações possíveis para faces triangulares são apresentadas no quadro V.17.

Aresta1	out	out	out	out	out	out	out	out	out
Aresta2	out	out	out	in	in	in	on	on	on
Aresta3	out	in	on	out	in	on	out	in	on
Resultado	out	erro	out	erro	erro	erro	out	erro	out
Aresta1	in	in	in	in	in	in	in	in	in
Aresta2	out	out	out	in	in	in	on	on	on
Aresta3	out	in	on	out	in	on	out	in	on
Resultado	erro	erro	erro	erro	in	in	erro	in	in
Aresta1	on	on	on	on	on	on	on	on	on
Aresta2	out	out	out	in	in	in	on	on	on
Aresta3	out	in	on	out	in	on	out	in	on
Resultado	out	erro	out	erro	in	in	out	in	on

Quadro V.17 – Classificação das faces em função de suas arestas

V.3.4.4 – AVALIAÇÃO BOOLEANA E ELIMINAÇÃO DOS COMPONENTES INDESEJADOS

Depois que todas as entidades topológicas dos objetos A e B tiverem sido classificadas, a avaliação booleana torna-se simplesmente uma tarefa de decidir quais entidades devem ser mantidas e quais destruídas. Todas elas foram classificadas em relação a ambos os objetos A e B, e recebem agora uma das oito combinações possíveis. O objeto do qual o elemento originalmente veio, recebe sempre a classificação *on*. Assim, os elementos de A recebem uma das classificações *onAinB, onAonBshared, onAonBanti-shared, onAoutB,* enquanto os elementos de B são classificados como *inAonB, onAonBshared, onAonBantishared* ou *outAonB.* Todas essas possibilidades apresentam-se nos dois objetos da figura V.30a, estando destacadas as classificações de ciclo mais importantes. Como só existem oito classificações possíveis, a ação apropriada para o algoritmo de avaliação booleana pode ser facilmente tabulado, como apresentado no quadro V.18.

Para a operação A – B, a intenção é manter todas as partes de A que estão somente em A e descartar todas as partes de A que também estão em B assim como todas as partes exclusivas de B. Isso é o que está colocado no quadro. Os elementos classificados como *inAonB* possuem a ação "manter/inverter": a superfície deve ser retida porque ele se torna parte da nova fronteira entre A e o resto do mundo, porém a superfície normal existente aponta para dentro do sólido A, refletindo o fato de que esta superfície foi originalmente parte exterior do sólido B. Assim, a normal à superfície deve ser invertida. O resultado obtido ao realizar a subtração, nos dois exemplos, é mostrado na figura V.30b.

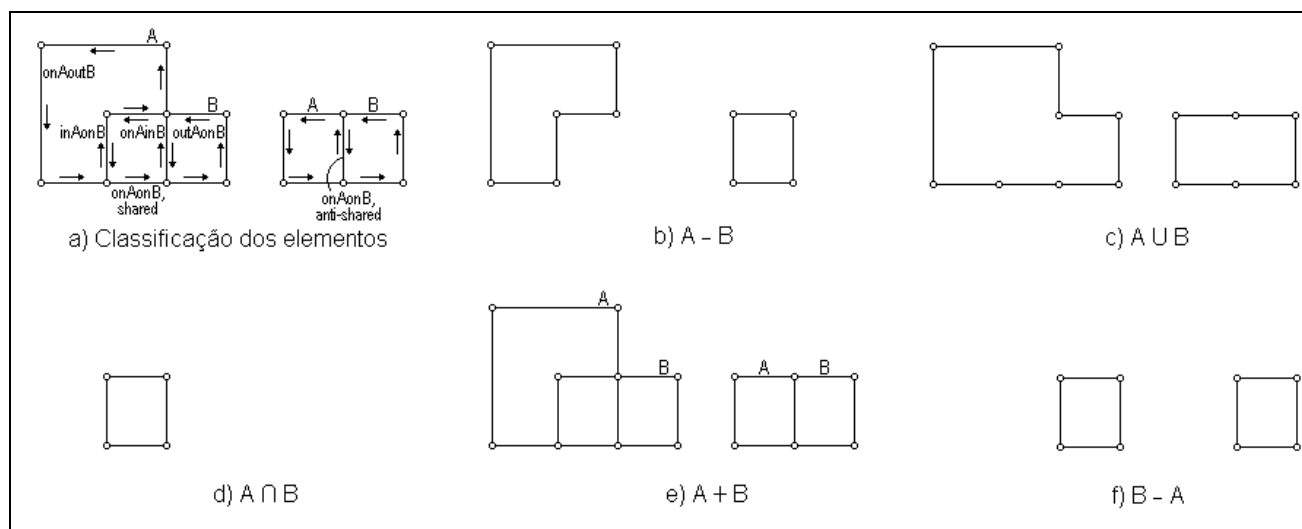



Figura V.30 – Classificação e avaliação das operações booleanas (baseado em [Muu91])

A	B	A – B	A ∪ B	A ∩ B	A + B	B – A
on	*in*	descartar	descartar	manter	manter	manter / inverter
on	*on shared*	descartar	manter	manter	manter	descartar
on	*on anti-shared*	manter	descartar	descartar	manter	manter
on	*out*	manter	manter	descartar	manter	descartar
in	*on*	manter / inverter	descartar	manter	manter	descartar
on shared	*on*	descartar	descartar	descartar	descartar	descartar
on anti-shared	*on*	descartar	descartar	descartar	descartar	descartar
out	*on*	descartar	manter	descartar	manter	manter

Quadro V.18 – Tabela de decisão das operações booleanas (baseado em [Muu91])

Para a união, a intenção é manter todos os elementos que estão no exterior de A ou de B e descartar qualquer estrutura interna ou elemento redundante. Mais precisamente, para a modelagem de sólidos, a fórmula da união pode ser interpretada como $A \cup B = (A - B) + (B - A) + (B \cap A)$, sendo + uma operação de soma simples. O resultado da união é ilustrado na figura V.30c.

A interseção, mostrada na figura V.30d, retém todos os elementos que são, simultaneamente, parte de A e de B, descartando as duplicações. A montagem, operação introduzida por necessidade específica deste trabalho, retém todos os elementos, inclusive as fronteiras internas, como ilustrado na figura V.30e. Finalmente, a diferença $B - A$ é apresentada, seguindo a mesma tática de $A - B$ mas com as referências a A e B invertidas.

A técnica de avaliação booleana e eliminação de componentes acima descrita é razoavelmente simples de programar e depurar, trabalha de forma confiável e é de fácil compreensão. O armazenamento direto de todas as regras de decisão em uma tabela torna o algoritmo bastante direto. Já as etapas anteriores de cálculo de interseções entre triângulos, resultando em novos triângulos, e de classificação dos componentes são bastante complexas. Elas envolvem diversas situações particulares que merecem cuidados especiais, os quais demandariam um tempo maior de pesquisa, o que levou à não implementação das operações booleanas neste trabalho.

O próximo passo consistiria em melhorar a malha obtida, unindo triângulos coplanares muito pequenos que compartilham segmentos comuns – o que reduziria o espaço de armazenamento utilizado – ou regularizando a triangulação, pelo deslocamento de um nó para o baricentro do polígono definido pelo conjunto de triângulos que o compartilham, como ilustrado na figura I.5 (capítulo I). Vencidas todas essas etapas da avaliação da fronteira, uma representação triangularizada, consistente e não redundante do resultado *A op B* seria obtida e poderia ser utilizada como operando em uma próxima operação da estrutura CSG.

"Concentre-se em fazer coisas melhores, não maiores."
H. Jackson Brown Jr.

VI – O SUBSISTEMA DE GERAÇÃO DE MALHA

O Subsistema de Geração de Malha cuida da geração da malha superficial de elementos finitos, derivada da representação B-rep e abre caminho para a geração da malha volumétrica, que é obtida pela tetraedrização dos volumes delimitados pela malha superficial. Diferentemente de outros, neste modelador o processo de geração da malha superficial inicia-se juntamente com a criação das primitivas. Assim, ao aplicar as operações booleanas, a malha inicialmente gerada sobre as primitivas é aproveitada, agilizando o processo de obtenção da malha final. Este subsistema não só divide os componentes em elementos que respeitam as fronteiras internas e externas do modelo, mas também controla a inclusão e o posicionamento de nós auxiliares, de forma a garantir a compatibilidade e a qualidade da malha resultante. Visando à etapa de processamento, cuida também da atribuição de características físicas e condições de contorno aos elementos geométricos e de malha que compõem o modelo.

VI.1 – TÉCNICAS DE GERAÇÃO DE MALHA UTILIZADAS COMO REFERÊNCIA

O processo de geração de malha utilizado neste modelador segue uma estratégia semelhante à proposta por Kamel e Chen [Kam91]. Na estratégia por eles definida, sobre cada primitiva é gerada uma malha utilizando qualquer técnica de geração de malha disponível, como a propagação frontal da malha ou a triangulação de Delaunay. Essas primitivas são então combinadas entre si por meio das operações booleanas de união, interseção e diferença, e uma nova malha é obtida com base na malha inicial definida sobre as primitivas, como ilustra a figura VI.1. Como a malha superficial gerada sobre a primitiva servirá de base para a malha superficial do objeto CSG resultante, é importante garantir a qualidade da malha sobre a primitiva para que a malha do objeto resultante também possua boa qualidade.

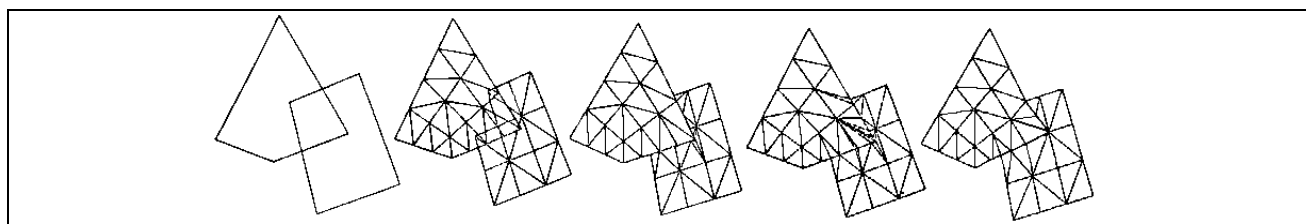



Figura VI.1 – Exemplo da técnica CSG com geração de malha sobre primitivas [Kam91]

A estratégia para geração da malha superficial sobre primitivas, utilizada neste modelador, está baseada em uma técnica para a discretização de esferas bastante conhecida, descrita a seguir. Além de triangularizar blocos, cilindros, cones, pirâmides, prismas e esferas, ela pode ser utilizada sobre qualquer objeto definido por varredura rotacional ou translacional, como aqueles que possuem simetria axial e são topologicamente equivalentes a esferas ou toros.

A técnica para discretização da superfície de uma esfera parte da subdivisão de um octaedro nela inscrito. Cada triângulo do octaedro está abaixo da superfície esférica, com seus três vértices exatamente sobre essa superfície. Ao criar pontos médios na lateral de cada triângulo, é possível criar quatro novos triângulos contidos no triângulo anterior. Ao projetar estes pontos sobre a superfície esférica, obtém-se uma melhor aproximação triangularizada dessa superfície. Esse processo de subdivisão, esquematizado na figura VI.2a, permite refinar a malha da esfera de maneira recursiva, até que se obtenha um nível de discretização adequado para o problema em questão. A figura permite observar ainda que, ao aplicar essa técnica recursiva a partir do ponto superior do triângulo, a cada linha horizontal de definição dos triângulos é acrescido um vértice. Cada uma destas linhas horizontais corresponde a uma "curva de nível" na esfera – o caminho percorrido pelos vértices que compõem o perfil gerador da esfera. Como cada triângulo inicial do processo recursivo corresponde a um octante da esfera, a cada curva de nível ocorre a inclusão de quatro vértices. Ao completar um hemisfério, o número de vértices por curva de nível começa a decrescer na mesma proporção, até atingir novamente um único vértice, perfazendo, assim, o restante da esfera. Essa propriedade de aumento e decremento do número de nós por curva de nível permite estender o processo de discretização da esfera para qualquer superfície gerada por varredura translacional ou rotacional.

Existe uma tendência a considerar o algoritmo de subdivisão triangular descrito para a esfera como sendo adaptativo, ou seja, capaz de apresentar áreas de maior curvatura com um maior refinamento da malha. Infelizmente, não é possível usar diferentes níveis de subdivisão sem introduzir "quebras" na triangulação, como a que é vista na figura VI.2b. Ao mesmo tempo que a "quebra" poderia ser evitada dividindo-se o triângulo da forma apresentada, isso iria produzir uma irregularidade na topologia da triangulação, o que tornaria o processo não recursivo e significativamente mais difícil.

a) recursividade na discretização de uma esfera b) subdivisão adaptativa

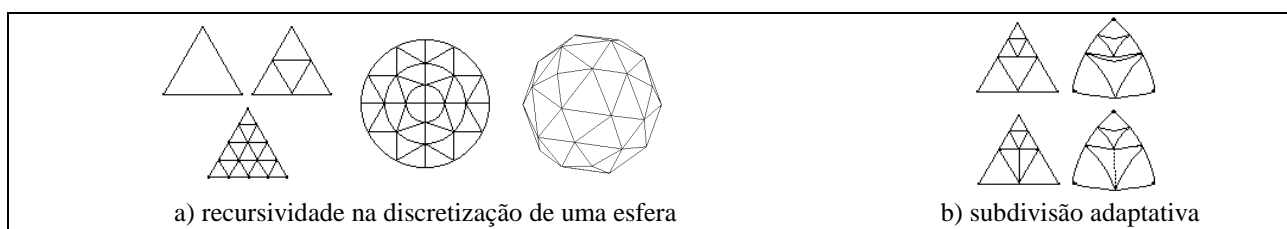



Figura VI.2 – Recursividade na discretização da esfera e problemas com a subdivisão adaptativa

VI.2 – PRINCIPAIS CARACTERÍSTICAS DE PROJETO

O projeto do Subsistema de Geração de Malha englobou basicamente o processo de geração de malha superficial sob primitivas, incluindo os casos de varredura translacional e rotacional, além da discretização de faces planares, necessárias para completar objetos definidos por varredura. A especificação de requisitos procurou levantar as principais características necessárias para que a malha resultante possua boa qualidade. Considerando os casos a serem tratados, os assuntos foram facilmente identificados, e a partir deles foram definidas as principais classes envolvidas. Alguns aspectos foram um pouco mais detalhados, investigando soluções orientadas para objetos, a serem adotadas na implementação.

VI.2.1 – REQUISITOS RELACIONADOS À GERAÇÃO DE MALHA

O Subsistema de Geração de Malha é responsável por criar a malha superficial sobre todos os componentes do modelo, a partir da representação de sua fronteira, e derivar desta a malha volumétrica, obtida pela tetraedrização dos volumes delimitados pela malha superficial. A técnica a ser empregada deverá iniciar a geração da malha superficial tão logo as primitivas sejam definidas, de forma que a malha gerada sobre elas possa ser aproveitada pelas operações de modelagem posteriormente aplicadas, o que deverá agilizar o processo de obtenção da malha final. Ao discretizar primitivas e manipular malhas, este subsistema deverá gerar elementos que respeitem as fronteiras internas e externas do modelo, bem como controlar a inclusão e o posicionamento de nós auxiliares, de forma a garantir a compatibilidade e a qualidade da malha resultante.

Elementos triangulares de primeira ordem deverão ser gerados sobre todas as faces do modelo. Para assegurar a qualidade da malha resultante, deve ser criado o mínimo possível de ângulos obtusos, uma vez que eles degradam a precisão dos resultados. Segundo Colyer et al. [Col97], para gerar malha sobre superfícies não planares é necessário o seguinte procedimento: projetar as faces em um espaço paramétrico 2D; definir uma malha sobre essa projeção, utilizando um dos algoritmos de geração de malha 2D descritos na literatura; re-projetar a malha para o seu espaço original, levando em consideração distorções geradas pela conversão entre estes espaços. No estágio atual do modelador, esse procedimento não será seguido, uma vez que as faces derivadas de primitivas serão geradas sempre a partir do processo de varredura, que aproximará faces curvas por facetas triangulares planas.

Vários requisitos devem ser considerados durante a geração de uma malha superficial, conforme expõe Boender et al. [Boe94]. Em primeiro lugar, a malha resultante deverá ser topológica e geometricamente correta, ou seja, os elementos deverão possuir uma topologia bem definida e ajustar-se corretamente à fronteira, sem interceptá-la. No caso do modelador, isso significa também que os nós sobre a aresta comum a duas faces da fronteira de um objeto devem pertencer a essas duas faces e conter os vértices do modelo B-rep, garantindo a compatibilidade entre os vértices da fronteira e os nós da malha superficial. Em segundo lugar, os nós da fronteira devem ser posicionados exatamente nas arestas e faces do modelo de forma que, no limite máximo de refinamento da malha, ela corresponda exatamente ao modelo geométrico. Além disso, para que a malha resultante possua boa qualidade, ela deverá conter o mínimo possível de elementos mal formados e possuir maior densidade onde o gradiente da função que está sendo aproximada for maior. Para atender a este último requisito, o modelador deverá incorporar algum mecanismo que permita especificar a densidade local para a malha superficial resultante, de forma que regiões que necessitem maior refinamento na malha possam ser obtidas e controladas pelo usuário. A densidade local da malha poderá ser obtida pela subdivisão das arestas que delimitam as faces, sendo essa subdivisão definida pela atribuição de pesos para um ou ambos os nós da aresta.

A partir da malha superficial triangularizada, o Subsistema de Geração de Malha deverá gerar a malha volumétrica, aplicando algoritmos de tetraedrização segundo o método de Delaunay, conforme descrito por Gonzalez [Gon98, Gon00].

Além de discretizar o domínio, o Subsistema de Geração de Malha deve também gerenciar a atribuição de características físicas e condições de contorno aos elementos geométricos e de malha que compõem o modelo, completando as informações a serem passadas para a etapa de processamento, que empregará o método de elementos finitos para resolver o problema representado pelo modelo.

VI.2.2 – O Modelamento Obtido

Diferentemente do que ocorre com os demais, os assuntos relacionados a este subsistema foram definidos em função dos casos de geração de malha superficial a serem tratados – varredura rotacional de perfis abertos ou fechados, varredura translacional simples ou cônica e discretização de faces planares – estando todos eles englobados em um assunto mais amplo, denominado Malha Superficial, ilustrado na figura VI.3b. A esse assunto associam-se os referentes à Atribuição de *Label* e Especificação de Materiais, conforme detalhado na figura VI.3a.

a) assuntos relacionados ao subsistema b) detalhamento do assunto Malha Superficial

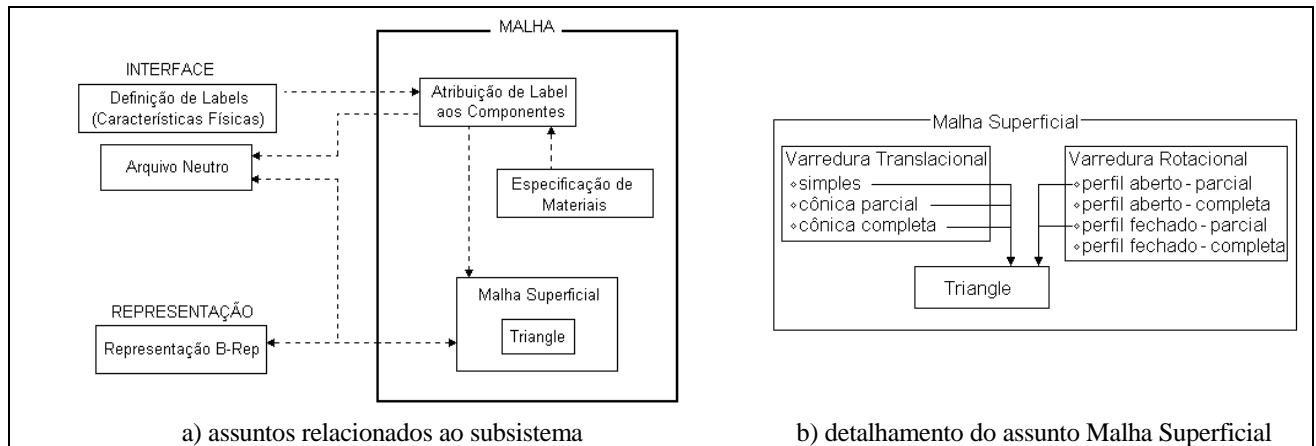



Figura VI.3 – Identificação de assuntos relacionados ao Subsistema de Geração de Malha

Como já comentado anteriormente, a Malha Superficial é gerada a partir da fronteira definida pela Representação B-rep. Fornecida por intermédio da Interface, a Especificação de Materiais integra-se à composição dos *Labels*, que são atribuídos aos componentes da Malha Superficial. Informações geométricas da Representação B-rep unem-se às informações da Malha Superficial e geram o Arquivo Neutro, onde também são descritas as condições de contorno e as características físicas e de material definidas nos *Labels* atribuídos aos componentes.

As classes envolvidas foram derivadas dos assuntos identificados, estando as principais relacionadas no quadro VI.1 juntamente com seus atributos e métodos mais importantes. As relações de dependência entre elas encontram-se esquematizadas na figura VI.4

O assunto Especificação de Materiais gerencia as características físicas referentes a materiais do modelo, por meio da classe *Materials*. Uma vez definido, um material pode ser associado a um componente do modelo por meio de um *label*.

O assunto Atribuição de *Label* relaciona um *label* contendo características físicas e condições de contorno a elementos do modelo B-rep – Região, Casca, Face, Aresta ou Vértice – e/ou a elementos da malha. Quanto à definição de *label*, refere-se às classes *Label*, *AttDefinitions*, *Atribute*, *AttNode*, *Materials* e *Bconditions*, descritas no capítulo III. A atribuição de *label* inclui ainda a interface, que controla a seleção dos elementos que receberão os *labels* definidos.

Mais abrangente que os anteriores, o assunto Malha Superficial envolve as classes *RotSweep* e *TranSweep*, responsáveis pelos processos de varredura, que geram não só a malha superficial, mas também a seqüência de operadores de Euler que permitem construir a representação B-rep de cada primitiva definida. Envolve ainda as classes *PolFile*, *Triangle*, *Border* e *Mesh*, que geram, sobre uma face poligonal planar e com o auxílio do software *Triangle* [She96], a malha superficial e a representação B-rep correspondente.

CLASSE: DESCRIÇÃO	
PRINCIPAIS ATRIBUTOS	PRINCIPAIS SERVIÇOS
PolFile: manipula arquivo composto por polilinhas que descrevem faces contendo ou não buracos	
nome do arquivo, ponteiros para sua manipulação, identificador e coordenadas dos vértices, vértice inicial de cada polilinha, pontos internos para identificação de buracos, identificador inicial para novos vértices, orientação	ler e gravar dados no arquivo, retornar dados
Triangle: encapsula a função *worktriangle (argument)* do programa *Triangle*	
função *worktriangle (argument)*	ativar função *worktriangle*, com diferentes argumentos, que lê um arquivo encapsulado pela classe PolFile e apresenta os resultados em arquivos
Border: vetor circular dinâmico que mantém o controle da fronteira (borda) sobre a qual a malha está sendo gerada	
vetor contendo os identificadores dos vértices, na seqüência em que estão definidos na fronteira controle para tornar esse vetor circular	adicionar, subtrair, substituir, modificar e retornar elemento, localizar e retornar próximo elemento, contar ocorrências de elemento, procurar por seqüência de elementos, retornar anterior e próximo
Mesh: cria a seqüência de Operadores de Euler que definem a estrutura B-rep para a malha definida sobre faces planares	
identificadores da região, da casca, da face e dos vértices B-rep correspondentes, ponteiro para arquivos envolvidos, referência para a borda que está sendo manipulada, total de pontos manipulados, orientação adotada, número e identificação de buracos, identificação e coordenadas dos vértices envolvidos, identificação e seqüência dos nós que formam cada elemento triangular gerado pelo Triangle	ler e gravar arquivos envolvidos, montar fronteira incluindo todos os buracos, identificar segmentos, nós e elementos, marcar elemento já processado, verificar se elemento já foi processado, atualizar borda, realizar todos os MEV possíveis a cada etapa, realizar todos os MEF possíveis a cada etapa
SweepPrim, Primitive2D, SweepOp, RotSweep, TranSweep: definição de primitivas por varredura	
Primitiva 2D e operação de varredura aplicada (obs: classes já definidas no subsistema de modelagem)	gerar malha sobre primitivas definidas por varredura
NeutralFile: arquivo neutro a ser gerado com os dados do modelo atual	
Nome, especificação do modelo, especificação do arquivo, matrizes auxiliares para montagem dos dados (obs: mesma classe já definida no subsistema de interface)	Gerar arquivo, gravar cabeçalhos do arquivo, gravar registros de dados
Label: definição dos *labels* contendo características físicas	
Nome, referência à lista de atributos (obs: mesma classe já definida no subsistema de interface)	Definir, obter e gravar atributos
AttDefinitions: contém todas as tabelas de atributos definidas	
Todas as tabelas de atributos definidas	Gerenciar tabelas: adicionar, obter, alterar atributos
Attributes: atributo pertencente à lista de atributos	
Tipo, identificador (obs: mesma classe já definida no subsistema de interface)	Definir, obter e gravar tipo e identificador
AttNode: definição de um atributo qualquer (genérico) – corresponde a uma linha da tabela de atributos	
Tipo, vetor de valores, referência a uma lista de pontos (obs: mesma classe já definida no subsistema de interface)	Homogeneizar tratamento de atributos
Materials: especificação de um material utilizado	
Nome, referência a uma lista de atributos (obs: mesma classe já definida no subsistema de interface)	Definir, obter e comparar atributos
SatPoint: especificação de pontos de saturação	
Identificador, valores de H e B	Definir, obter e comparar atributos
AttBCNode:	
Tipo, valores e label associados (obs: mesma classe já definida no subsistema de interface)	Definir, obter e comparar atributos
BConditions: especificação de uma condição de contorno	
Nome, lista de atributos da condição de contorno (obs: mesma classe já definida no subsistema de interface)	Definir, obter e comparar atributos
TEditTablesForm, TAppendBoundaryConditionsForm, TAppendMaterialForm e demais formulários para definir *labels*	
Campos para digitação e seleção de características físicas	Realizar a entrada de dados para atributos relacionados às características físicas de *labels*

Quadro VI.1 – Principais características das classes relacionadas

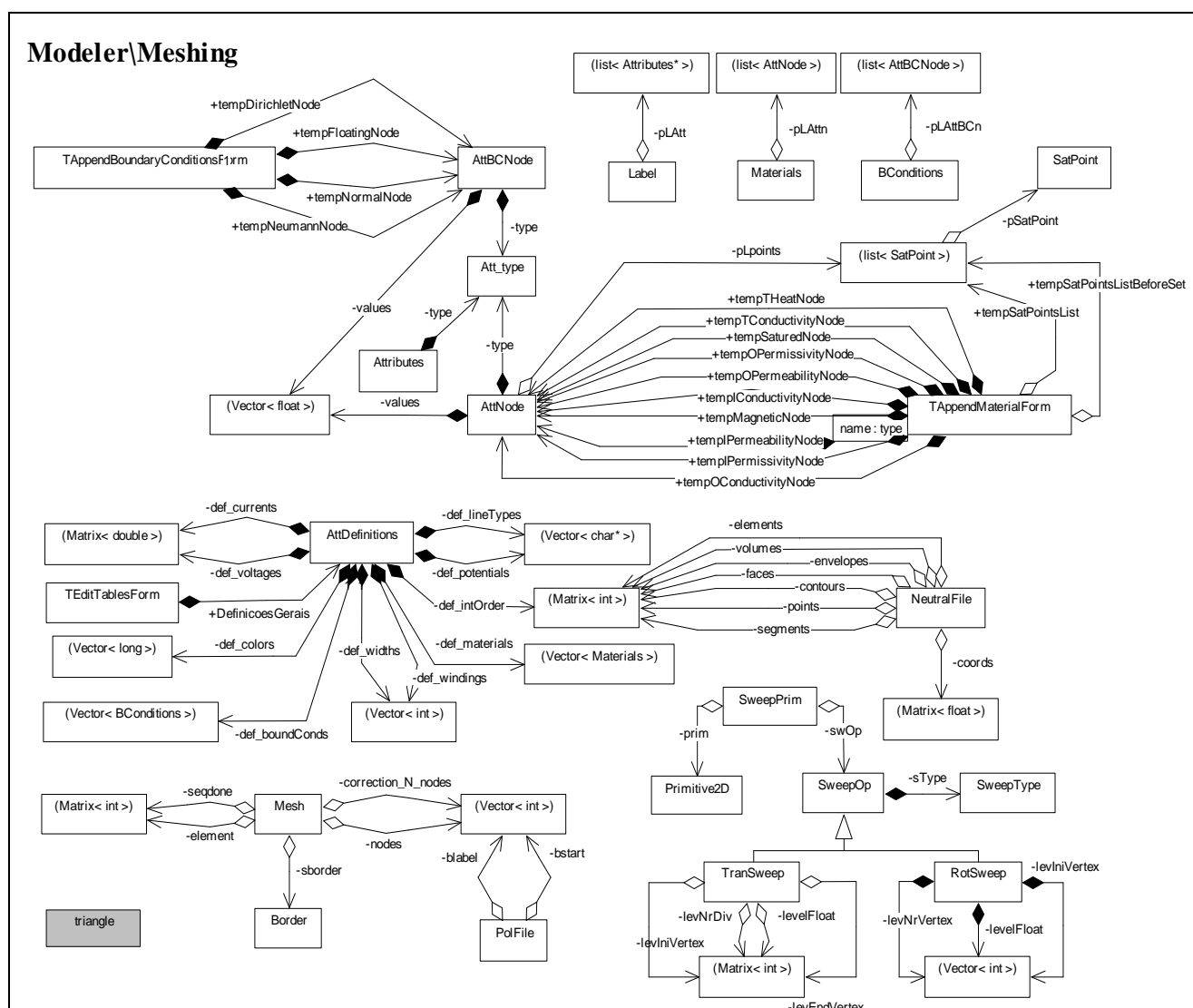



Figura VI.4 – Diagrama das principais classes e relações de dependência entre elas

VI.2.3 – SOLUÇÕES ORIENTADAS PARA OBJETOS

Para objetos definidos por varredura, uma característica peculiar apresentada pelo Subsistema de Geração de Malha é a obtenção da malha superficial pela composição da malha gerada sobre as faces laterais com a produzida sobre o perfil gerador. A malha gerada sobre as faces laterais é obtida a partir dos processos de varredura translacional ou rotacional, detalhados nos itens VI.3.1 e VI.3.2. Já a malha gerada sobre as faces planares, definidas sobre o perfil gerador após a execução da varredura rotacional parcial ou translacional, é obtida utilizando o programa *Triangle*, como explicado no item VI.3.3. Esse subsistema é responsável pela garantia da compatibilidade entre essas malhas, bem como por sua junção.

VI.3 – Principais Características de Implementação

A implementação do Subsistema de Geração de Malha iniciou-se juntamente com a construção das operações de varredura translacional e rotacional, pertencentes ao Subsistema de Modelagem, já que a malha deve ser gerada sobre as superfícies definidas pela varredura. Iniciou-se, em seqüência, a geração de malha sobre superfícies planares, uma vez que primitivas obtidas pela varredura rotacional parcial ou translacional criam faces planares não triangularizadas, definidas sobre o perfil gerador.

VI.3.1 – A Geração de Malha para a Varredura Rotacional

VI.3.1.1 – A Geração de Malha para a Varredura Rotacional de um Perfil Aberto

A esfera pode ser vista como um caso especial de varredura rotacional, em que um arco – o perfil – é varrido sobre uma trajetória circular. O processo de discretização da esfera, apresentado na seção VI.1, pode então ser estendido para qualquer objeto gerado por varredura rotacional, desde que se mantenha a propriedade de aumento e decremento do número de nós por curva de nível. A aplicação do processo para um perfil genérico possui algumas facilidades: pode ser aplicado para varreduras envolvendo qualquer ângulo $\leq 360^0$ e, ao invés de considerar a divisão em octantes ($= 90^0$), podem ser definidos vários setores de mesma angulosidade ($\leq 120^0$). O número de setores deve ser escolhido em função de um tamanho máximo de segmento a ser utilizado. Quanto menores forem esses setores, maior será o refinamento da malha resultante. A partir do vértice inicial, que é único, o número de vértices – ou nós – por curva de nível cresce ou decresce de *s*, número de setores circulares.

Por outro lado, a geração de malha sobre um perfil genérico exige alguns cuidados: os nós utilizados no processo devem estar distribuídos sobre o perfil gerador de maneira que o número total de acréscimos e decréscimos de nós seja o mesmo. Além disso, uma discretização homogênea requer que os nós sejam distribuídos de forma a gerar elementos cujo formato se aproxime, ao máximo, de triângulos equiláteros, sendo necessário, portanto, fazer com que os segmentos tenham o comprimento mais homogêneo possível.

Para homogeneizar o comprimento dos segmentos de um perfil contendo os pontos inicial e final definidos sobre o eixo de rotação, calcula-se inicialmente o comprimento de cada segmento do perfil, ou seja, a distância entre cada par de nós consecutivos contidos no perfil. A partir destes valores obtém-se o comprimento médio de todos os segmentos. A seguir, estabelece-se o número de subdivisões a serem realizadas sobre esse comprimento médio, definindo um referencial para comprimento – o segmento padrão. Os segmentos do perfil maiores que esse padrão serão divididos em partes iguais,

cujo tamanho é o mais próximo possível do segmento padrão, mas nunca o ultrapassa, como é mostrado na vista frontal da figura VI.5a. Nós adicionais são incluídos a cada divisão realizada.

Uma vez calculado o segmento padrão e redefinido o perfil, um outro procedimento é aplicado para gerar os nós de cada setor da superfície varrida. Esse procedimento parte do nó inicial do perfil, contido no eixo de rotação, e define n_v (número de nós de cada curva de nível naquele setor) em função de n_s (número inteiro de segmentos padrão que podem estar contidos na distância do nó ao eixo de rotação): se n_s exato, $n_v = n_s + 1$; senão $n_v = n_s + 2$. Por estarem sobre o eixo de rotação, o nó inicial e o final definem sozinhos o seu nível. Como o comprimento dos segmentos do perfil redefinido é sempre menor ou igual ao do segmento padrão, a variação do número de nós entre duas curvas de nível consecutivas é sempre *0* ou *1*. Isso pode ser observado na vista superior da figura VI.5a, que mostra os nós definidos e suas respectivas curvas de nível, além de uma curva de nível auxiliar (curvas de nível auxiliares são círculos concêntricos cujos raios são múltiplos do segmento padrão).

Se ao percorrer o perfil seguindo a seqüência de seus segmentos ocorrer a passagem por uma curva de nível auxiliar, o número de nós deverá ser incrementado – caso o percurso esteja se afastando do eixo – ou decrementado – caso esteja aproximando-se do eixo. Se não ocorrer a passagem por uma curva de nível auxiliar, o número de nós deverá ser mantido. A figura VI.5b reproduz a malha de um setor destacando o número de nós por nível. As coordenadas dos nós do setor são facilmente obtidas em função da distância do nó ao eixo de rotação e do número de nós contidos na curva de nível do setor. Uma visão da malha gerada e o processo de desenho são apresentados na figura VI.5c. Nota-se que o setor fica aberto de um dos lados, o que permite a repetição do algoritmo para os demais setores, até que seja completada a varredura, quando o último setor deverá ser unido ao primeiro setor definido (se ângulo = 360^0), ou fechado (se ângulo < 360^0). A definição dos nós dos outros setores é obtida pelo incremento do ângulo referente ao setor, seguido de novo cálculo de coordenadas.

a) vistas do perfil b) malha obtida c) traçado e visão da malha

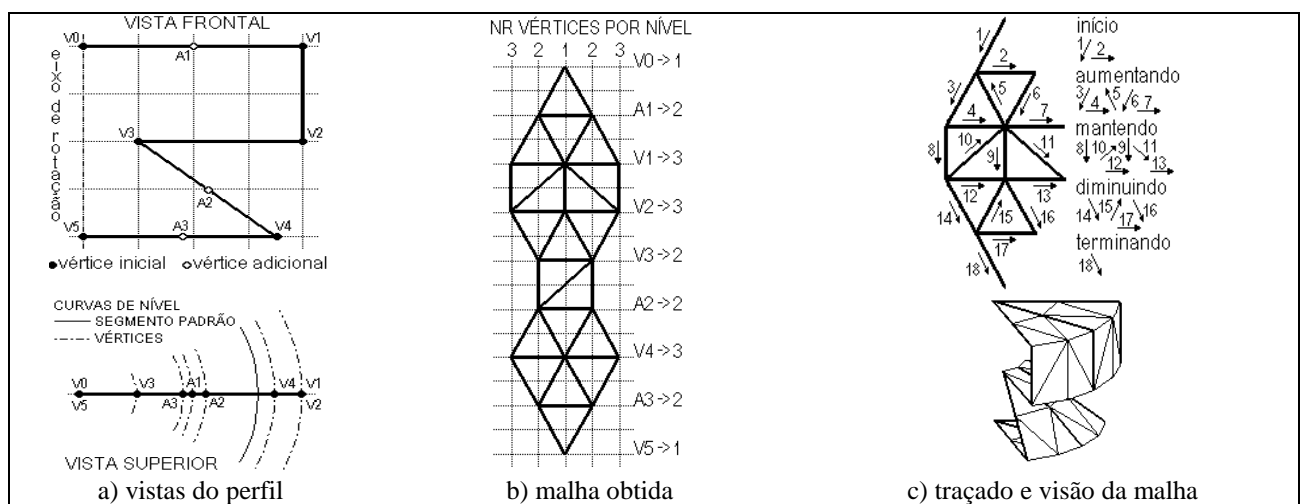



Figura VI.5 – Exemplo de varredura rotacional de um perfil aberto

Ao invés de traçar um setor por vez, este procedimento pode ser realizado, opcionalmente, por curva de nível definida, uma vez que todos os setores de uma mesma curva de nível estão, simultaneamente, iniciando, aumentando, mantendo, diminuindo ou terminando. A seqüência de traçado por curva de nível é detalhada na figura VI.6. A figura VI.7 apresenta alguns exemplos de malhas obtidas a partir da varredura rotacional de perfis abertos.

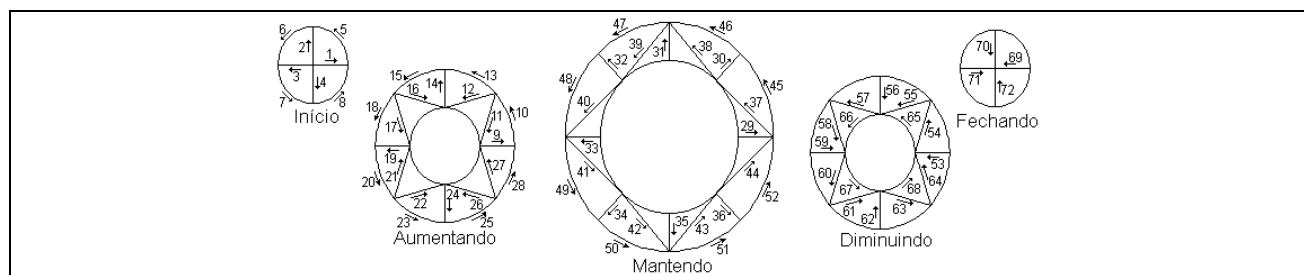



Figura VI.6 – Exemplo de traçado por nível da varredura rotacional de um perfil aberto

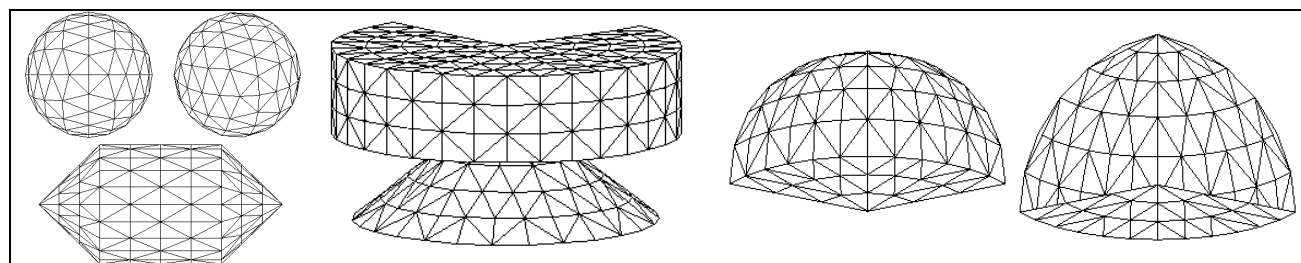



Figura VI.7 – Exemplos de malhas obtidas pela varredura rotacional de um perfil aberto

VI.3.1.2 – A GERAÇÃO DE MALHA PARA A VARREDURA ROTACIONAL DE UM PERFIL FECHADO

A mesma semelhança entre a varredura rotacional de perfil aberto e a esfera pode ser encontrada na varredura rotacional de um perfil fechado e o toro. Objetos gerados por este tipo de varredura são topologicamente semelhantes (homeomorfos) a um toro. O princípio de geração utilizado para a construção de um toro pode assim ser generalizado para a varredura fechada.

Um toro pode ser visualizado da forma ilustrada na figura VI.8a, descrita por Muuss e Butler [Muu91]: se um de seus lados for cortado transversalmente e o toro for esticado, ele se tornará um cilindro circular reto. Se este cilindro for cortado longitudinalmente e descurvado, ele se tornará um retângulo que possui um lado com comprimento igual ao perímetro do círculo obtido com o corte transversal (percorrido à medida que o ângulo α varia de 0 a 2π) e o outro lado com comprimento igual à distância em torno da seção longitudinal do toro (percorrida à medida que o ângulo β varia de 0 a 2π). Este toro poderia também ser definido por um círculo de raio $r1$ em α varrido em torno de β, com raio $r2$.

Alguns comentários importantes podem ser feitos em relação à malha do toro: se fosse realizada uma discretização homogênea sobre a superfície retangular, como mostra a primeira opção da figura VI.8b, e seu resultado fosse rebatido sobre o toro, a malha obtida não seria de boa qualidade, uma vez que resultariam triângulos muito pequenos na parte interior e triângulos muito grandes na parte exterior. Uma alternativa seria determinar o número de segmentos a serem usados para aproximar os contornos internos e externos, distribuir nós sobre o contorno – como mostra a segunda opção da figura VI.8b – e aplicar um dos métodos de geração de malha apresentados na literatura, como a propagação da fronteira e o método de Delaunay, resumidos no capítulo I. Apesar de mais genérica, esta abordagem implicaria um maior tempo computacional.

a) decomposição do toro b) opções para a geração de malha

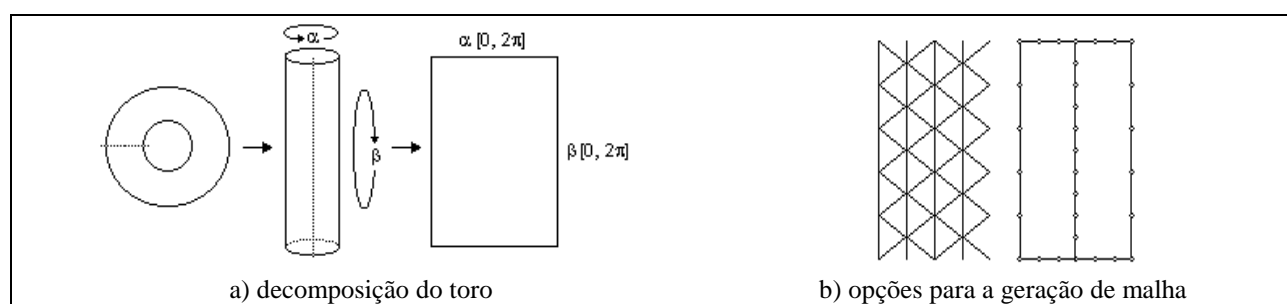



Figura VI.8 – Decomposição de um toro e possíveis discretizações geradas sobre a decomposição

Uma forma alternativa de obter uma malha seria utilizar um procedimento semelhante ao apresentado para a esfera. Curvas de nível poderiam ser definidas para nós simetricamente posicionados na circunferência que compõe o perfil a ser varrido. O processo se iniciaria com um número reduzido de nós no círculo central – um nó definindo cada setor. Um incremento fixo de um nó por setor a cada nível seria então empregado, até completar a metade superior do toro e, a seguir, para a metade inferior do toro, um processo semelhante decrementaria um nó a cada setor. Não obstante esse processo permitir obter um toro com elementos de tamanho mais homogêneo, a qualidade da representação no círculo interno ainda não seria boa, como mostra a figura VI.9a. O uso de uma concentração um pouco maior de nós no círculo interno levaria a um crescimento exagerado do número de nós no círculo externo, o que ocorre devido ao estabelecimento de um valor fixo para o incremento e o decremento de nós a cada nível.

Um solução para aumentar a qualidade do círculo interno sem sobrecarregar de elementos o círculo externo é refinar o procedimento anterior, permitindo aumentar, manter ou diminuir o número de nós em cada nível, em função de um valor de referência, como ocorre com o segmento padrão definido para a varredura rotacional de um perfil aberto, apresentado na seção anterior. Nesse procedimento, a homogeneização do comprimento dos segmentos ocorre da mesma forma: o

comprimento médio de todos os segmentos é calculado; seu valor é dividido de forma a obter o segmento padrão; os segmentos do perfil são redivididos de forma que o comprimento de cada um se aproxime do segmento padrão, sem ultrapassá-lo. A seguir, o número de nós de cada nível é estabelecido, suas coordenadas são calculadas e a malha do setor é traçada. O processo é repetido para o setor seguinte, até que todos os setores, menos o último, tenham sido traçados. Para a varredura completa (360^0), os nós do último setor deverão ser unidos aos seus correspondentes no primeiro setor. Já para a varredura parcial (ângulo < 360^0), os nós deverão ser fechados entre si seguindo a seqüência definida pelo perfil gerador. Exemplos fio-de-arame desta abordagem são apresentados na figura VI.9b.

a) aplicando o princípio da esfera, com incrementos e decrementos fixos

b) incrementando, mantendo ou decrementando em função da posição da curva de nível, controlada pelo segmento padrão

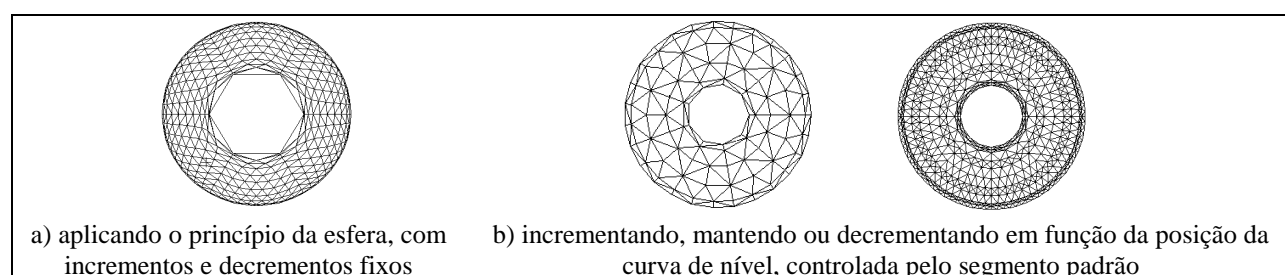



Figura VI.9. Formas de geração de malha utilizando a varredura rotacional de um perfil fechado

O procedimento para traçado da malha por curva de nível definida segue uma abordagem semelhante à apresentada para a varredura rotacional de perfil aberto. A curva de nível escolhida para o início do traçado é a que está mais próxima do eixo de rotação. As etapas de início e fechamento possuem algumas alterações significativas devido à existência do anel central. Para o início, apenas o anel deverá ser traçado. Para o fechamento, será necessário definir seqüências de traçado diferentes para os casos em que o número de nós se mantém ou decresce. Como o procedimento parte da curva mais próxima do eixo, nunca ocorrerá um acréscimo do número de nós no fechamento. A seqüência de traçado por curva de nível é detalhada na figura VI.10. Este procedimento pode ser utilizado para qualquer perfil fechado que defina uma face sem buracos ou interseções. Alguns exemplos ilustrativos de varredura rotacional de perfil aberto e fechado são apresentados na VI.11.

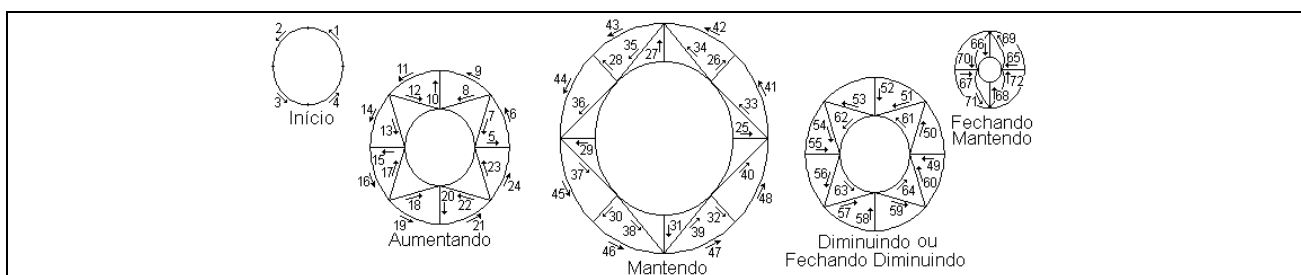



Figura VI.10 – Exemplo de traçado por nível da varredura rotacional de um perfil fechado

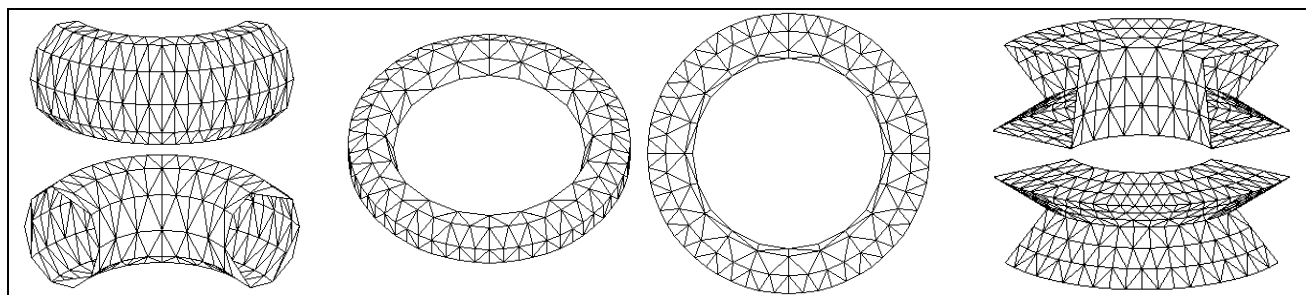



Figura VI.11 – Exemplos de malhas obtidas pela varredura rotacional de um perfil fechado

VI.3.1.3 – A GERAÇÃO DOS OPERADORES DE EULER A PARTIR DA VARREDURA ROTACIONAL

A construção da representação B-rep correspondente às varreduras rotacionais de perfil aberto e de perfil fechado segue a mesma seqüência definida em seus traçados reproduzidos, respectivamente, nas figuras VI.6 e VI.10. O operador inicialmente aplicado na geração de qualquer primitiva é o MVFR, que define o modelo esquelético composto por uma região, uma face e um vértice. A partir dele, todos os vértices são adicionados ao modelo por meio do operador MEV, o qual acrescenta a aresta que liga o vértice ao restante da representação. Todas as faces são fechadas pelo operador MEF, que também adiciona uma aresta à representação. Duas observações são de fundamental importância na estratégia definida: durante todo o processo, a face manipulada é sempre a inicial, o que facilita o fornecimento de parâmetros para a execução dos operadores. Os elementos geométricos já adicionados à representação estão sempre conectados entre si, e qualquer novo elemento incorporado deverá estar de alguma forma conectado aos demais.

Assim como ocorre com o traçado, a seqüência de operadores definida pela varredura rotacional de um perfil aberto é diferente da definida pela varredura de um perfil fechado, mas pode ser gerada por nível ou por setor. Em busca de maior eficiência, optou-se pela geração por nível, com a seqüência de operadores repetindo-se a cada setor. Grosso modo, a geração dos operadores obtida pelos dois processos é diferente apenas no nível inicial e no fechamento, conforme mostra o quadro VI.2. Para os setores inicial e final, as semi-arestas passadas como parâmetro aos operadores variam bastante, conforme a varredura seja parcial ($< 360^o$) ou completa ($= 360^o$). A varredura completa de um perfil fechado é a que apresenta maior variação na seqüência de operadores empregados: no início, deve gerar o buraco interno da primitiva de *genus* 1 – homeomorfa a um toro – o que é realizado pelo operador KFMLH e, no final, deve incorporar esse buraco à face de trabalho, por meio do operador MEKL. A geração dos operadores de Euler para a varredura rotacional encontra-se implementada no módulo *RotSweep*.

Quadro VI.2 – Seqüência de operadores de Euler gerada em cada etapa da varredura rotacional

VI.3.2 – A GERAÇÃO DE MALHA PARA A VARREDURA TRANSLACIONAL

VI.3.2.1 – A GERAÇÃO DE MALHA PARA A VARREDURA TRANSLACIONAL SIMPLES

Como dito anteriormente, objetos gerados por varredura translacional simples são obtidos transladando um perfil gerador pelo caminho definido por um vetor diretor não coplanar ao perfil. Se o deslocamento for perpendicular ao plano de definição do perfil, obtém-se uma translação reta, caso contrário, obtém-se uma translação oblíqua. As faces laterais serão sempre compostas por quatro arestas, formando retângulos para a translação reta e paralelogramos para a translação oblíqua. As arestas do perfil gerador definem o lado do quadrilátero contido no plano de definição do perfil – geralmente horizontal –, enquanto o vetor deslocamento define o outro lado.

Uma boa malha para uma translação reta pode ser gerada utilizando o traçado retangular básico, exposto na figura VI.12a. Cada aresta do perfil gerador é dividida em segmentos de igual tamanho, levando em consideração o segmento padrão estabelecido, como apresentado na seção anterior. Este mesmo procedimento é utilizado para dividir o lado definido pelo vetor deslocamento. Cada segmento paralelo ao perfil gerador corresponde a uma curva de nível anteriormente definida. Como o comprimento de cada segmento do perfil gerador mantém-se a cada nível da translação, a abordagem utilizada em cada passo corresponde à estratégia "mantém" apresentada na varredura

rotacional. A seqüência de traçado também é a mesma. Vale a pena destacar, porém, a inversão ocorrida no traçado diagonal em cada nível, visando evitar uma direção preferencial para a malha, o que poderia gerar erros em sua aplicação pelo método de elementos finitos. Tal inversão deve ser suprimida em varreduras translacionais oblíquas que possuam um ângulo entre o vetor deslocamento e o plano gerador menor que 60^0 ou maior que 120^0, pois nesses casos a inversão piora a qualidade da malha, como mostra a figura VI.12b. Para tais casos, basta inverter o traçado que leva à formação de triângulos obtusângulos, favorecendo a formação de triângulos próximos a equiláteros, como mostra a figura VI.12c. A nova seqüência de traçado também é apresentada nessa figura. A figura VI.13 mostra alguns exemplos de malha gerada sobre superfícies definidas por varredura translacional simples.

a) traçado retangular básico   b) mesmo traçado inclinado   c) alteração para melhoria da malha

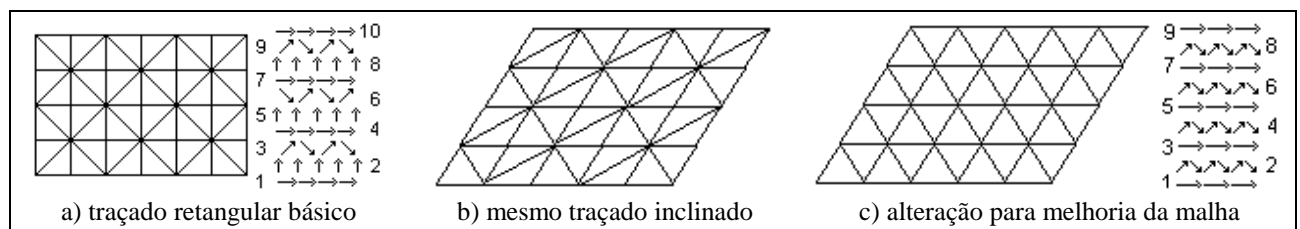



Figura VI.12 – Traçado de malha utilizado na varredura translacional simples

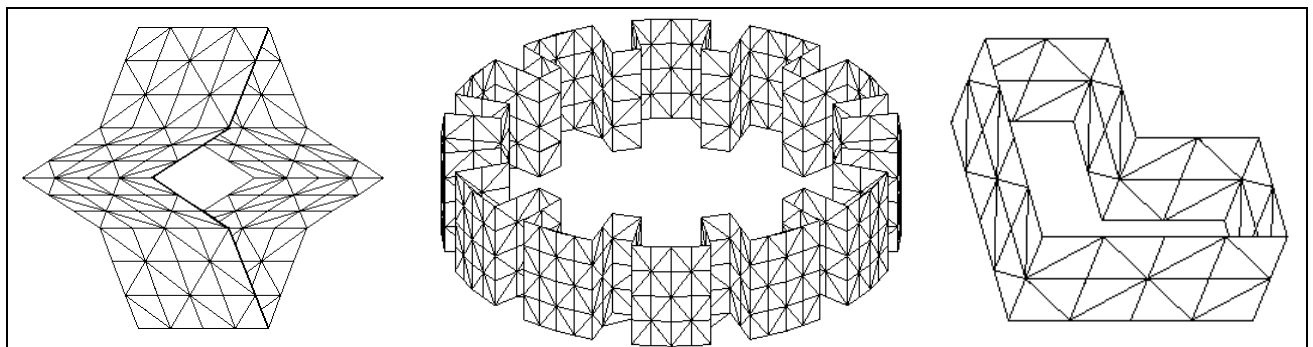



Figura VI.13 – Exemplos de malhas obtidas pela varredura translacional simples

VI.3.2.2 – A GERAÇÃO DE MALHA PARA A VARREDURA TRANSLACIONAL CÔNICA

A geração de malha para a varredura translacional cônica também segue a mesma idéia, porém é um pouco mais complicada que a varredura translacional simples. A variação no comprimento do segmento leva a uma variação no número de segmentos por curva de nível, resultando em uma combinação entre níveis que contêm um mesmo número de elementos e níveis que diferem em um segmento. A figura VI.14a apresenta o traçado triangular básico, utilizado na discretização da esfera, que diminui um segmento por nível. Seguindo a mesma abordagem e as considerações da varredura translacional simples, a figura VI.14b mostra o traçado utilizado para manter o número de segmentos

por nível. É importante notar que deverá ser analisado o ângulo contido em cada lado do triângulo para, a partir daí, definir a direção do traçado diagonal, visando sempre à obtenção de triângulos acutângulos e próximos ao equilátero. A figura VI.14c mostra a combinação entre estes dois traçados.

a) traçado triangular básico — b) traçado ao manter nr pontos — c) combinação de traçados

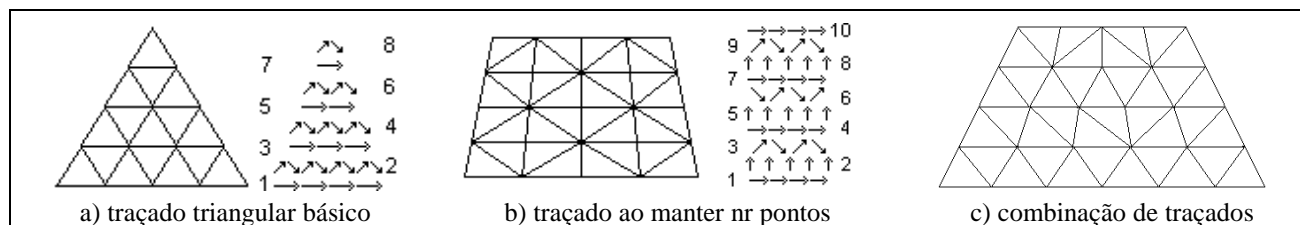



Figura VI.14 – Traçado utilizado na varredura translacional cônica

Além destas considerações em relação a cada face gerada pela varredura, observa-se que o número de segmentos entre as arestas do perfil pode variar, levando à possibilidade de existir, em um mesmo nível, faces que mantêm e faces que decrementam o número de segmentos. No exemplo da figura VI.15a, a face lateral sempre decrementa, enquanto na figura VI.15b ocorre uma variação não linear do número de segmentos entre os níveis. A geração da malha, portanto, deverá ser independente para cada segmento do perfil, porém o número de níveis, bem como o comprimento dos segmentos definidos sobre o vetor diretor devem manter-se constantes, de forma a garantir a compatibilidade entre os segmentos do perfil, como se vê na figura VI.15c. Alguns exemplos ilustrativos de varredura translacional cônica parcial e completa são apresentados na figura VI.16.

a) face lateral: lado maior — b) face lateral: lado menor — c) combinação obtida na varredura

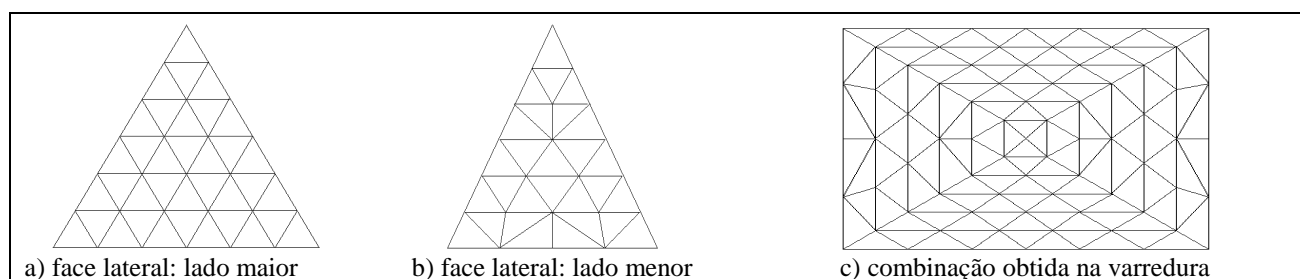



Figura VI.15 – Malha resultante da varredura translacional cônica de um perfil retangular

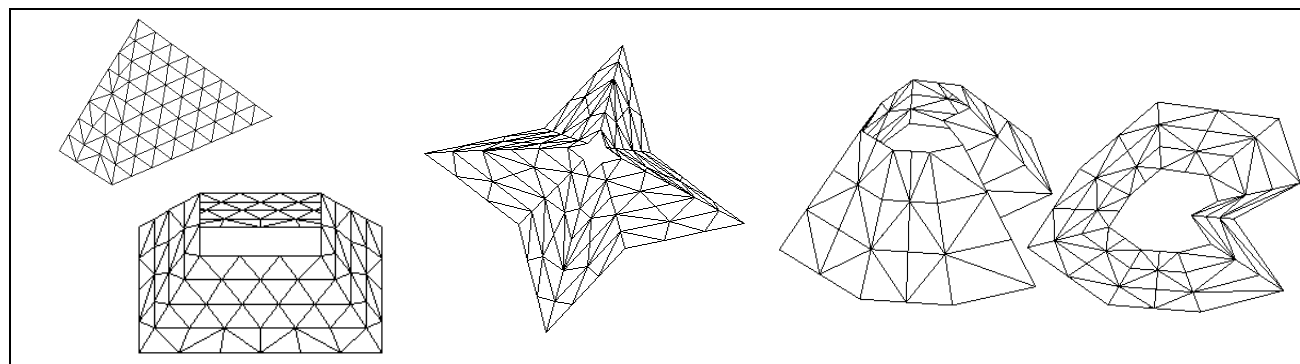
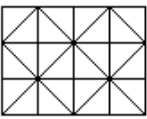
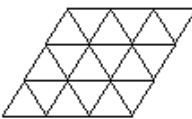
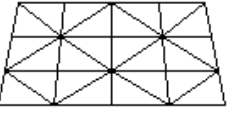
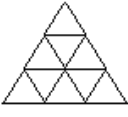



Figura VI.16 – Exemplos de malhas obtidas pela varredura translacional

VI.3.2.3 – A GERAÇÃO DOS OPERADORES DE EULER A PARTIR DA VARREDURA TRANSLACIONAL

A construção da representação B-rep correspondente às varreduras translacionais – simples e cônica – segue a mesma seqüência definida em seus traçados, reproduzidos nas figuras VI.12 e VI.14, respectivamente. A seqüência de operadores de Euler gerada pela varredura translacional reta corresponde à apresentada pela estratégia "mantém" da varredura rotacional. Varreduras translacionais simples oblíquas em que o valor do ângulo formado pelo vetor deslocamento e o plano gerador está compreendido entre 60^0 e 120^0 devem sofrer uma inversão no traçado diagonal em cada nível, o que altera as semi-arestas passadas como parâmetro para operador MEF, empregado no fechamento das faces. Na varredura translacional cônica, deverá ser analisado o ângulo contido em cada face lateral triangular por ela gerada, a fim de definir a direção do traçado diagonal. Além da estratégia "mantém" com inversão de traçado onde necessário, a varredura translacional cônica segue a estratégia "diminui" da varredura rotacional, gerando a mesma seqüência de operadores de Euler. O quadro VI.3 mostra os operadores resultantes em cada caso.

Quadro VI.3 - Seqüência de operadores de Euler gerada em cada caso da varredura translacional

VI.3.3 – A GERAÇÃO DE MALHA SOBRE FACES PLANARES

Em modelos gerados por varredura rotacional parcial ou translacional, além das superfícies resultantes da varredura, fazem parte do modelo as faces planares definidas pelo perfil gerador. Sobre essas faces deverá também ser gerada uma malha superficial triangularizada, compatível com a malha lateral obtida pela varredura. Como a face definida pelo perfil gerador pode assumir qualquer forma, para sua triangulação é utilizado o método de Delaunay, por ser bem mais genérico. A geração de malha nessas faces é realizada com o auxílio do programa *Triangle*. A malha superficial resultante para uma primitiva é, portanto, a união da malha superficial obtida pela varredura com a malha superficial gerada pelo programa *Triangle*.

VI.3.3.1 – O PROGRAMA *TRIANGLE*

O *Triangle* é um programa para geração de malha bidimensional, desenvolvido em linguagem C pela *Carnegie Mellon University* [She96]. Ele utiliza o método de Delaunay para triangularizar faces poligonais, que podem conter ou não buracos. Os pontos pertencentes à triangulação de Delaunay possuem a propriedade de não estarem contidos no interior de nenhum círculo formado por três vértices que compõem um triângulo resultante do processo.

O *Triangle* recebe como entrada um grafo planar orientado, denominado PSLG – *Planar Straight Line Graph* –, definindo regiões planares limitadas por segmentos de reta que podem conter ou não buracos. A malha resultante possui obrigatoriamente os segmentos definidos no grafo. A triangulação forçada – *constrained Delaunay triangulation* – não permite a divisão desses segmentos, que continuarão a ser representados como arestas únicas após a triangulação. Como resultado, obtém-se uma pseudo-triangulação de Delaunay, uma vez que poderá não ser mais satisfeita a propriedade anteriormente citada. Já a triangulação adaptada – *conforming Delaunay triangulation* – permite particionar os segmentos originais em várias arestas pela inserção de pontos adicionais, incluídos não só para satisfazer restrições quanto a ângulo mínimo e/ou área máxima, aprimorando a malha resultante, mas também para satisfazer a propriedade anterior, gerando uma triangulação de Delaunay autêntica e adequada para a análise por elementos finitos.

A figura IV.17 apresenta um grafo e as malhas resultantes de cada uma das estratégias anteriores. O *Triangle* procura sempre reduzir o número de triângulos, de forma que as simulações não tomem mais tempo que o necessário. Se for necessário simular fenômenos físicos com maior precisão, triângulos pequenos podem ser definidos, alterando a geometria do grafo PSLG ou limitando a área ou o ângulo dos triângulos a serem gerados, o que é definido pelas opções e parâmetros de execução.

a) grafo PSLG   b) triangulação forçada   c) triangulação adaptada (sem e com restrição de área)

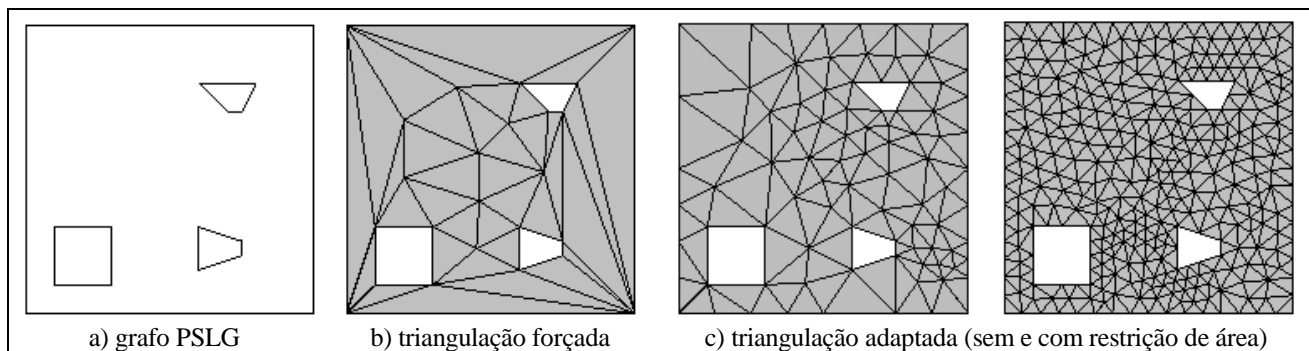
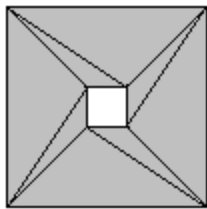



Figura VI.17 – Variações de malha que podem ser obtidas pelo programa *Triangle*

O programa *Triangle* recebe como entrada um grafo PSLG, que é definido por um arquivo com extensão ".POL", cujo formato está descrito no quadro VI.4. Todos os arquivos manipulados pelo *Triangle* podem conter comentários, prefixados pelo símbolo '#'. Linhas e espaços adicionais são desprezados. A numeração adotada pode começar de 0 ou 1, mas deve ser seqüencial e consistente, independentemente de seu valor inicial: se os vértices são numerados a partir de 1, todas as numerações devem iniciar em 1. O *Triangle* detecta automaticamente essa escolha ao ler um arquivo ".POL".

Antes de chamar o *Triangle*, o modelador gera um arquivo ".POL" contendo um grafo PSLG ideal, isto é, sem interseção de segmentos nem pontos definidos sobre eles – exceto, obviamente, seus próprios pontos extremos. Além do grafo, o arquivo ".POL" gerado pelo modelador possui informações específicas, relativas a definições de buracos e referências a vértices do modelo, que são colocadas logo após o símbolo de comentário, acompanhadas de "$" ou outro "#", como mostrado no exemplo do quadro VI.4.

Após o processamento, o *Triangle* gera arquivos com extensão ".ELE" e ".NOD". O arquivo ".ELE" descreve os elementos da malha, ou seja, a seqüência de nós referente a cada triângulo gerado, o que especifica indiretamente a orientação do elemento. As coordenadas correspondentes a cada nó estão descritas no arquivo ".NOD", acompanhadas do identificador do nó. O formato desses arquivos é apresentado no quadro VI.5. Opcionalmente, o *Triangle* pode gerar também outros arquivos, não utilizados pelo modelador, entre eles: ".NEI", que indica a vizinhança dos elementos; ".EDG", que define características para arestas; ".ARE", que associa a cada triângulo uma área máxima, a ser usada no refinamento da malha pelo *Triangle*.

ARQUIVOS ".POL"

```
1a linha: <# de pontos> <dimensão(2)> <# de atributos>
    <# de marcações de fronteira>
demais linhas: <# ponto> <x> <y>         => obs: usados
    [atributos] [marcador de fronteira]  => obs: não usados
uma linha: #$ <# de buracos> <ini1> <fim1>...<inin> <fimn>
uma linha: <# de segmentos> <# de buracos>
demais linhas: <# segmento> <# ponto inicial> <# ponto final>
uma linha: <# de buracos>
demais linhas: <# buraco> <x> <y> => obs: qq. ponto no buraco
0   obs: marcador de fim de arquivo para o Triangle
uma linha: ## <# pontos do arquivo +1> <r1>...<rn> <r_ini>
    obs: <r1>...<rn> relacionam os rótulos dos pontos no
    modelador e <r_ini> é o primeiro rótulo disponível, a
    partir do qual novos pontos gerados deverão ser rotulados
```



```
#box.pol – quadrado com buraco
8 2 0 0
   1 0 0
   2 10 0
   3 10 10
   4 0 10
   5 4 4
   6 6 4
   7 6 6
   8 4 6
#$ 1 5 8
8 1
   1 1 2
   2 2 3
   3 3 4
   4 4 1
   5 5 6
   6 6 7
   7 7 8
   8 8 5
1
   1 5 5.2
0
## 9 21 22 23 24 33 34 35 36 50
```

Quadro VI.4 – Formato dos arquivos com extensão ".POL"

ARQUIVOS ".NOD" 1a linha: <# de pontos> <dimensão(deve ser 2)> <# de atributos> <# marcações de fronteira (0 ou 1)> demais linhas: <# ponto> <x> <y> [atributos] [marcador de fronteira] obs.: [atributos] e [marcador de fronteira] não são usados pelo modelador	# arquivo ".nod" correspondente 8 2 0 0 1 0 0 2 10 0 3 10 10 4 0 10 5 4 4 6 6 4 7 6 6 8 4 6
ARQUIVOS ".ELE" 1a linha: <# de triângulos> <pontos por triângulo> <# de atributos> demais linhas: <# triângulo> <#ponto> <#ponto> <#ponto> [atributos] obs.: <# ponto> é o definido no arquivo ".nod"; [atributos] não é usado pelo modelador	# arquivo ".ele" correspondente 8 3 0 1 1 5 8 2 5 1 2 3 4 8 7 4 8 4 1 5 6 2 3 6 2 6 5 7 7 3 4 8 3 7 6

Quadro VI.5 – Formato dos arquivos ".ELE" e ".NOD"

Para acoplar o programa *Triangle* ao modelador, foi necessário transformá-lo em uma função independente, denominada *meshFace*, que controla as opções de execução listadas no quadro VI.6, selecionadas entre as diversas disponíveis no *Triangle*:

```
Forma de chamada: triangle [-pq_a_BPYiQ] input_file
- p : realiza a triangulação a partir de um arquivo ".pol"
- q_: restrição de ângulo imposta; se valor não fornecido, assume valor padrão
- a_: restrição de área imposta; se valor não fornecido, gera com área máxima
- B : evita a inserção de marcadores de fronteira, desnecessários ao modelador
- P : não escreve novo arquivo ".pol" visando refinamento
- Y : não adiciona pontos à fronteira
- i : utiliza o método incremental ao invés do método "dividir para conquistar"
- Q : suprime ou não a geração do arquivo "triangle.log", contendo explicações
      sobre o processamento do programa Triangle
- input_file: nome do arquivo ".pol", gerado pelo modelador
```

Quadro VI.6 – Opções de execução do *Triangle* controladas pelo modelador

VI.3.3.2 – A GERAÇÃO DOS OPERADORES DE EULER A PARTIR DOS RESULTADOS OBTIDOS DO *TRIANGLE*

A malha gerada pelo programa *Triangle* deverá ser refletida sobre o modelo B-rep. Para que isso ocorra, é necessário construir a seqüência de Operadores de Euler correspondente à estrutura da malha e carregá-la na representação B-rep armazenada pelo Núcleo. De início, desenvolveu-se uma estratégia que gerava sucessivamente, a partir da borda, todos os elementos triangulares da malha. Essa estratégia, porém, não impunha uma ordem para a geração dos triângulos, e muitas vezes a borda, que deveria manter-se única, acabava sendo dividida em várias – como ilustra a figura VI.18 – o que complicava todo o processo.

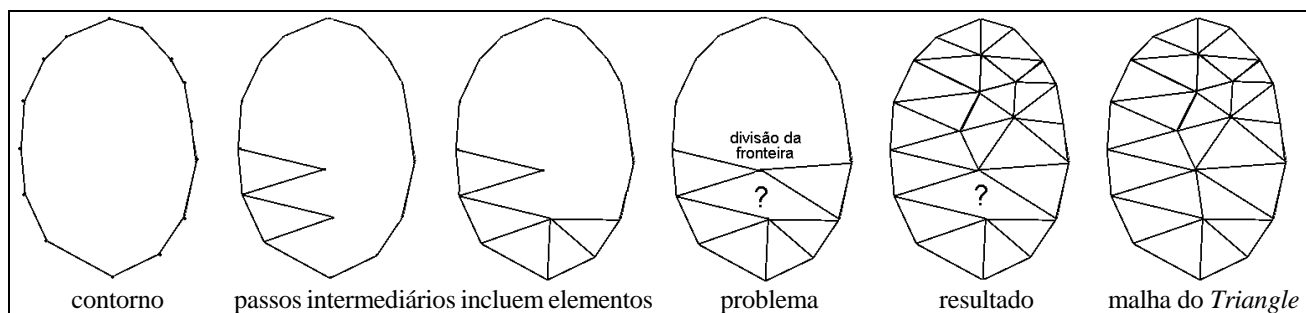



Figura IV.18 – Estratégia inicial para definir operadores de Euler a partir da malha gerada pelo *Triangle*

Verificou-se desta forma que o mais interessante seria trabalhar sempre com a mesma borda, o que levou ao desenvolvimento de uma nova estratégia. Ao invés de fechar elementos, optou-se por aumentar a borda, ligando a ela diversos pontos vizinhos, por meio do operador MEV. A seguir, um operador MEF é aplicado a todos os elementos que podem ser fechados, diminuindo novamente o número de segmentos da borda. Essas etapas são sucessivamente aplicadas, até que todos os pontos façam parte da borda e todos os elementos sejam gerados. A figura VI.19 elucida a estratégia empregada, que funciona para faces contendo ou não buracos internos.

Primitivas geradas por varredura translacional simples ou rotacional parcial possuem duas faces iguais contendo orientações opostas, que correspondem às faces inicial e final definidas sobre o perfil gerador. Essas faces deverão receber uma mesma malha definida pelo *Triangle*, mas sua representação B-rep deverá ser gerada com orientação invertida. Ao que tudo indica, a estratégia aqui descrita está preparada para gerar a representação B-rep nas duas orientações, a partir de qualquer malha definida pelo programa *Triangle*.

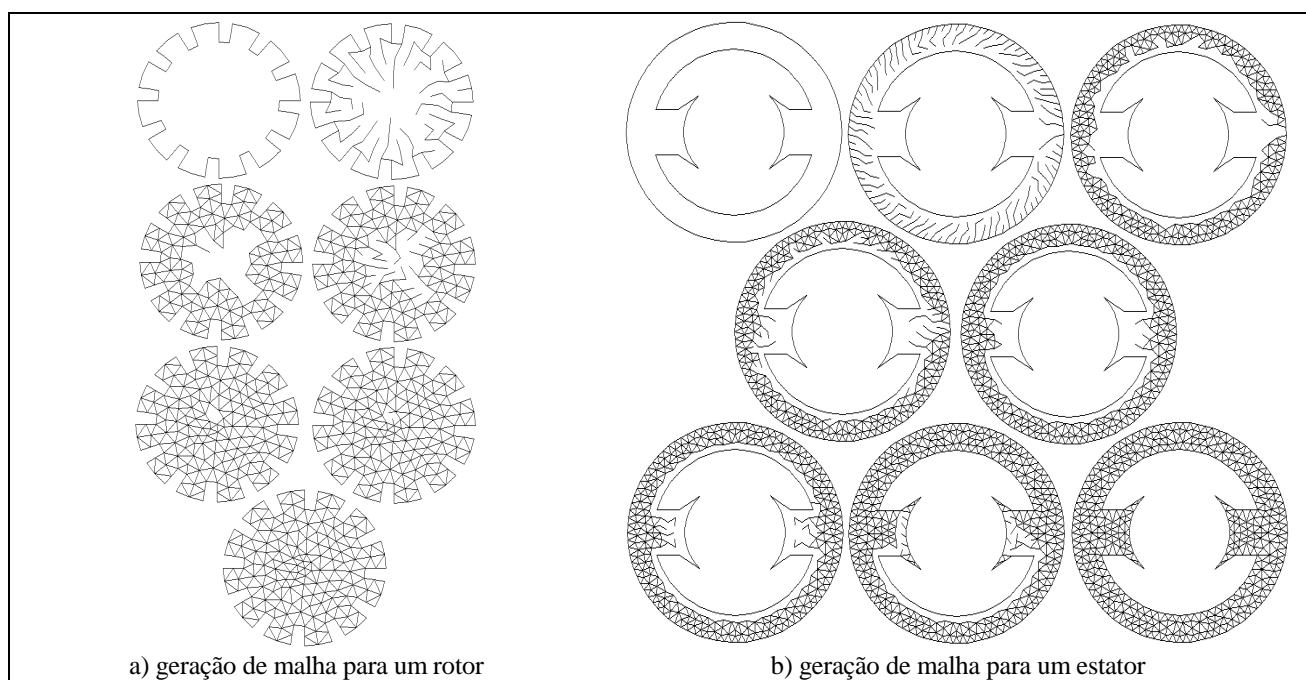



Figura IV.19 – Exemplos da estratégia para definir os operadores de Euler que incluem no B-rep a malha gerada pelo *Triangle*

VI.3.4 – Atribuição de *Label* e Especificação de Materiais

Um conjunto de atributos incluindo características físicas e dados relacionados ao cálculo pode ser vinculado aos elementos geométricos e de malha do modelo. Esse conjunto é referenciado por um *label*. Como todo elemento superficial ou volumétrico da malha possui atributos físicos, um *label* deve estar obrigatoriamente associado a ele. Opcionalmente, um *label* pode ser associado a cada um dos componentes da geometria, de ponto a objeto, bem como a nós e arestas da malha.

A figura VI.20 ajuda a explicar o mecanismo do *label*. Além de facilitar a identificação de componentes, um *label* permite atribuir ao componente características tais como: tipo de material, potencial para cálculo, condição de contorno ou valores de corrente e tensão impostas, enrolamentos e cor (definidas no arquivo de geometria e de malha); largura e padrão do traçado utilizado em sua representação geométrica (definidas no arquivo de geometria); ordem de integração e dados de carregamento (definidos no arquivo de malha). Além disso, um *label* permite cruzar as informações disponíveis: um componente geométrico pode ser especificado como atributo de um *label* para malha. Esta referência cruzada facilita a busca de informações sobre o modelo: por exemplo, ao incluir em um *label* para elemento superficial um atributo do tipo "face" – referência à face, definida no arquivo de geometria –, é possível saber com pequeno esforço computacional a qual envelope ou volume esse elemento pertence.

Além do funcionamento, a figura VI.20 mostra como foi implementado o mecanismo de *label*. Pode-se entender um *label* como uma referência para uma lista de identificadores de atributos, que são por sua vez definidos em tabelas contendo a descrição das características físicas e visuais utilizadas no modelo. Isto é o que ocorre para atributos como cor, corrente, potencial, tensão, enrolamento (usados tanto para geometria como para malha), largura e padrão de traçado (específicos da geometria) e ordem de integração (específico da malha). Como os atributos material e condição de contorno são, por sua vez, formados por um conjunto variável de novos atributos (ex: condutividade, permeabilidade e permissividade para materiais isotrópicos ou ortotrópicos; condição de contorno escalar – Dirichlet / Neumann – ou vetorial – normal / tangente), ao invés de utilizar uma tabela foi necessário definir uma lista contendo uma sub-lista de atributos para cada material ou condição definida, uma vez que não se sabe *a priori* quais atributos deverão ser utilizados em cada caso. O quadro VI.7 mostra a composição de dados de cada atributo, com a seguinte convenção: *[valor]* indica valor inteiro, <*valor*> indica valor real, *(nr)* indica string com *nr* caracteres. Os módulos *Label* e *Material* contêm a implementação dos mecanismos de atribuição de *label* e definição de material.

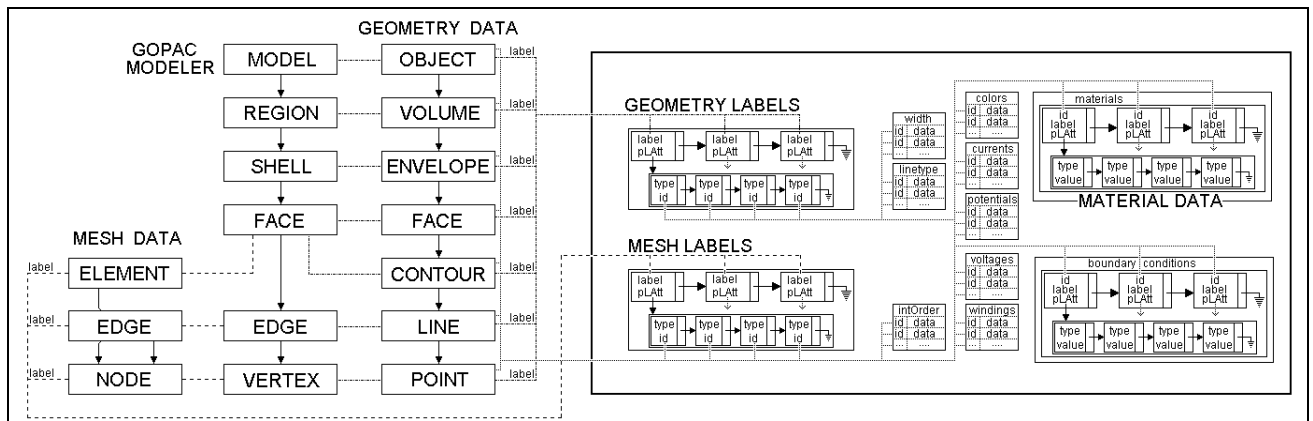



Figura VI.20 – Explicação do funcionamento e implementação do mecanismo de *label*

GEOMETRIA (width – material)
MALHA (color – Integration order)

ATRIBUTO	CONTEÚDO
width	[width_id] [#_of_pixels]
linetype	[linetype_id] 'LT_label(10)' onde o *label* de tipo de traçado pode ser: 'CONTINUOUS', 'DASHED', 'DOTTED', 'DASH-DOT'
color	[color_id] [R_value] [G_value] [B_value]
potential	[potential_id] 'potential_label(20)' onde o label para potencial pode ser: 'ELECTRIC_SCALAR', 'MAGNETIC_SCALAR', 'ELECTRIC_VECTOR', 'MAGNETIC_VECTOR', 'ELECTRIC_FIELD', 'MAGNETIC_FIELD', 'ELECTRIC_INDUCTION', 'MAGNETIC_INDUCTION', 'TEMPERATURE'. 'MAGNETIC_REDUCED'
current	[current_id] <J_real> <J_imag> <frequency>
voltage	[voltage_id] <voltage_real> <voltage_imag> <frequency>
winding	[winding_id] [#_of_coils]
boundary conditions	[BC_id] 'BC_label(20)' e mais uma lista de atributos definidos entre: BOUNDARY.SCALAR.DIRICHLET <Potential> BOUNDARY.SCALAR.NEUMANN <Density> BOUNDARY.VECTOR.NORMAL <A_1> <A_2> 'BC_vector_tangent_label(2)' onde o label de vetor tangente define em qual plano se encontra as componentes do vetor A. As componentes desse vetor são representadas por A_1 e A_2. Os labels podem ser: 'XY', 'XZ', 'YZ', 'RZ', 'TZ', 'RT', 'RF', 'TF', onde T=teta e F=fi. Ex: 1 A_x A_y 'XY' , 2 A□ A_z 'TZ', ... BOUNDARY.SCALAR.FLOATING não possui valor acoplado, apenas indica que componentes com este tipo terão mesmo potencial, independente de valor
material	[material_id] 'material_label(20)' e mais uma lista de atributos definidos entre: MATERIAL.PROPERTY.SATURATED.BH <m> e mais uma lista de pontos com densidade de fluxo e intensidade de campo [point_id] <H> <B> MATERIAL.PROPERTY.MAGNETIC <Bx> <By> <Bz> MATERIAL.PROPERTY.ISOTROPIC.CONDUCTIVITY <s> MATERIAL.PROPERTY.ISOTROPIC.PERMEABILITY <m> MATERIAL.PROPERTY.ISOTROPIC.PERMISSIVITY <e> MATERIAL.PROPERTY.ORTHOTROPIC.CONDUCTIVITY <sx> <sy> <sz> MATERIAL.PROPERTY.ORTHOTROPIC.PERMEABILITY <mx> <my> <mz> MATERIAL.PROPERTY.ORTHOTROPIC.PERMISSIVITY <ex> <ey> <ez> MATERIAL.PROPERTY.TERMIC.CONDUCTIVITY <k> MATERIAL.PROPERTY.TERMIC.HEAT <C>
Integration order	[integration_id] [order_r] [order_s] [order_t]

Quadro VI.7 Composição de dados de cada atributo utilizado para *labels*

VI.3.5 – A GERAÇÃO DA MALHA VOLUMÉTRICA

A malha superficial obtida é assim utilizada como ponto de partida para a geração da malha volumétrica pelo método de Delaunay. A discretização é construída iterativamente a partir da malha superficial triangularizada, dentro da qual é criada uma “nuvem” de nós – conjunto de pontos convenientemente distribuídos na fronteira e nas interfaces. A técnica de discretização utilizada consiste em conectar os nós existentes empregando a propriedade da tetraedrização de Delaunay [Geo91], obtendo uma malha tetraédrica cuja qualidade depende do número e localização dos nós que discretizam os contornos do modelo e do número e localização dos nós internos, compondo a nuvem de pontos.

Uma evolução que está sendo incorporada neste modelador é a geração de malhas adaptativas: partindo de uma malha grosseira, o procedimento obtém uma primeira solução, que permite detectar zonas onde o erro do método devido a deficiências da malha é maior. Nestas zonas são introduzidos novos pontos, a malha é refinada e a solução correspondente à nova malha é recalculada. Este processo continua até que o erro da geração da malha torne-se aceitável – homogeneizando a magnitude do erro em todo o volume de estudo – ou que o número máximo de elementos ou de nós seja atingido. Este processo está sendo desenvolvido em outra tese de doutorado no GOPAC [Gon98, Gon00].

"Os homens verdadeiramente grandes nunca separam o pensamento que defendem das ações que praticam."
Evanilde M. Dalcanale

VII – A GERAÇÃO DE MODELOS E DE BASE DE DADOS NEUTRA

Nos capítulos anteriores foram detalhados os recursos implementados para permitir a descrição, pelo usuário, de modelos CSG, bem como a construção, a partir desses modelos, de representações B-rep e de malhas de elementos finitos adequadas para aplicações eletromagnéticas.

Este capítulo apresenta exemplos de utilização, acompanhados de detalhes sobre a sua construção e a geração da base de dados neutra. Inicialmente, são discutidas as vantagens e desvantagens de utilizar-se uma base de dados neutra, comentam-se esforços internacionais em busca de uma padronização, apresenta-se a base de dados desenvolvida pelo GOPAC e mostra-se como ela está sendo gerada por este modelador. A seguir, são descritos alguns exemplos de composições simples obtidas com o modelador, acompanhados da seqüência de operadores de Euler aplicada para a obtenção do modelo, bem como da base de dados neutra correspondente. Finalizando, alguns exemplos mais complexos ilustram seu poder descritivo e sua aplicabilidade para a área de eletromagnetismo.

VII.1 - O INTERCÂMBIO DE DADOS E A GERAÇÃO DA BASE DE DADOS NEUTRA

Nenhum sistema de CAD é completo a ponto de atender a todos os usuários e aplicações. Para aplicações específicas, tornou-se necessário o desenvolvimento de sistemas específicos. Com a proliferação dos sistemas de CAD, o intercâmbio de dados entre eles tornou-se um requisito fundamental. Os usuários necessitam integrar sistemas distintos, procurando usufruir do potencial e da especificidade de cada um. Com o GOPAC não é diferente: com várias pesquisas em andamento e em etapas distintas, que incluem o desenvolvimento e a utilização de pré e pós-processadores para o cálculo de campos eletromagnéticos utilizando métodos diferentes, a integração de sistemas torna-se também muito importante, justificando o desenvolvimento e a utilização de uma base de dados contendo um formato neutro, a ser lido e gerado pelos diversos sistemas.

VII.1.1 - VANTAGENS E DESVANTAGENS DA UTILIZAÇÃO DE UMA BASE DE DADOS NEUTRA

Para o intercâmbio de dados entre dois sistemas diferentes, poderiam ser utilizados dois programas que fizessem a tradução direta e bidirecional entre esses sistemas, com a vantagem de ser possível adequar e otimizar esses programas para cada caso. Se essa mesma estratégia fosse, porém,

adotada para o intercâmbio de dados entre *n* componentes, *n(n–1)* programas tradutores seriam requeridos, e se outro componente fosse adicionado ao conjunto, *2n* programas tradutores adicionais deveriam ser desenvolvidos. Complicando ainda mais, para integrar diversos sistemas seria requerido um conhecimento detalhado sobre todos os formatos envolvidos, e qualquer alteração de formato acarretaria a necessidade de manutenção em todos eles.

Uma estratégia que soluciona o problema de intercâmbio de dados entre diversos sistemas é a utilização de um mecanismo neutro, capaz de descrever dados sobre o produto que está sendo modelado, independentemente de qualquer sistema específico. Tal mecanismo define um formato padrão pré-definido para dados e arquivos, permitindo organizar as informações sobre o produto de forma adequada, facilitando ao usuário sua geração e manipulação. A utilização de uma base de dados neutra é vantajosa, uma vez que *2n* programas são requeridos para *n* componentes e apenas *2* programas adicionais devem ser desenvolvidos para cada inclusão de componente. Além disso, o formato neutro padroniza a transferência de dados, melhorando a compatibilidade e a portabilidade dos dados e arquivos.

Por outro lado, não é possível conseguir que todas as características disponíveis em todos os sistemas sejam incorporadas pelo mecanismo neutro, e mesmo características incorporadas podem ser mal interpretadas se o formato padrão for incompleto ou ambíguo. Outro problema é que duas traduções ocorrem em cada transferência, e não uma, diminuindo a eficiência e dobrando a oportunidade de ocorrência de erros ou perda de informação. Tais desvantagens, porém, podem ser minimizadas se a cooperação entre os sistemas for considerada primordial e se estudos de viabilidade e testes de adaptação forem realizados *a priori*. O benefício obtido com a compatibilidade e a portabilidade de dados entre sistemas é muito maior do que qualquer uma das desvantagens apontadas.

VII.1.2 - Esforços Internacionais em Busca de uma Padronização

A necessidade de transferência de dados entre sistemas CAD emergiu no início dos anos 70, e a primeira especificação foi criada justamente para modeladores de sólidos a partir de um esforço voluntário [Wil87]. Na década de 80 houve uma proliferação de padrões de transferência de dados CAD, destacando-se o IGES, PDES, CAD*I [Kro89], até ocorrer um esforço combinado para o desenvolvimento de um padrão único, sob o patrocínio da ISO (*International Organization for Standardization*). Denominado ISO 10303 ou STEP (*Specification for the Transfer and Exchange of Product data*) [Owe97], este padrão internacional normatiza o intercâmbio de dados referentes ao

modelo de um produto a ser industrializado, permitindo acomodar várias representações diferentes, incluindo dados geométricos e de elementos finitos. O STEP pretende fornecer um mecanismo neutro, interpretável tanto pelo computador quanto pelo projetista, capaz de descrever computacionalmente dados sobre o produto durante todo o seu ciclo de vida, independentemente de qualquer sistema específico[Gu95]. De imediato, visa especificar um formato padrão de arquivo neutro e desenvolver pré e pós processadores para vários sistemas CAD comerciais, melhorando a compatibilidade e a portabilidade de entidades gráficas [Tho95, Kro89]. Uma proposta de normatização para os problemas eletromagnéticos está sendo discutida, visando atender a várias características específicas desses problemas [Sad93, Tho95].

Para a definição de um mecanismo neutro é necessário, primeiramente, selecionar e definir uma linguagem e uma estrutura básicas, de forma a permitir expressar entidades, propriedades, atributos e relações entre eles. Regras para a estrutura e sintaxe são fundamentais e devem ser adotadas por todos os programas e participantes do projeto. No STEP os modelos são definidos pela linguagem de especificação de dados (modelamento) denominada *Express* [ISO91]. Possui basicamente dois modelos de informação: um mais genérico, sobre recursos básicos (unidades, geometria, topologia, estrutura, etc.) e outro, mais específico, sobre protocolos entre aplicações, englobando uma descrição geométrica completa para visualização e análise dos resultados, bem como uma descrição simplificada, para entrada nos programas de cálculo.

A geração de arquivos neutros a partir de modeladores de sólidos – o pré-processamento – provou ser eficaz no STEP para casos em que foi necessário simplesmente relacionar todas as entidades que compõem o modelo, transmitido no formato apropriado para o arquivo neutro. Isso é possível quando todas as entidades do modelo possuem entidades correspondentes na especificação do arquivo e vice-versa, mas isso não é o que normalmente ocorre. Existem modeladores que possuem entidades não padronizadas, sem correspondente entre as entidades geométricas contidas na especificação do STEP. Nessas situações, ou o modelador deverá ser capaz de computar tais diferenças e adaptar-se ao arquivo neutro, ou esta função deverá ser de competência do pré-processador. Outro tipo de dificuldade é a inexistência, em um modelador, de certas entidades que precisam estar presentes no arquivo neutro.

A leitura de informação a partir de um arquivo neutro para um sistema CAD em particular – o pós-processamento – implica classificar a porção do arquivo que o sistema pode entender e tentar resolver o restante da melhor forma possível, considerando suas próprias características e restrições. O desenvolvimento de pós-processadores para o STEP tem-se mostrado mais difícil.

VII.1.3 – A BASE DE DADOS DESENVOLVIDA PELO GOPAC

Como a definição de um padrão internacional é muito lenta, e sentindo a necessidade de trocar informações durante a simulação eletromagnética de modelos, o GOPAC partiu para o desenvolvimento de uma base neutra própria [Roc95, Roc96, Sil96]. A base de dados neutra definida e utilizada pelo GOPAC permite integrar pré-processadores, processadores e pós-processadores em problemas relacionados ao cálculo de campos eletromagnéticos.

A base de dados neutra do GOPAC é basicamente composta por três arquivos: o de geometria, que contém a descrição geométrica do modelo, organizada em listas hierárquicas descrevendo pontos, linhas, contornos, faces, envelopes e volumes; o de malha, que contém a malha de elementos finitos descrita pelos nós, arestas, elementos superficiais e volumétricos que compõem o modelo, além de detalhes sobre o cálculo e condições de contorno; o de materiais, que contém os dados referentes às propriedades físicas dos materiais envolvidos no problema. Todos os arquivos são seqüenciais, orientados de modo linear e em formato ASCII, o que torna desnecessário o alinhamento de colunas e facilita a leitura por programas aplicativos e interfaces externas. Seu conteúdo está seccionado em blocos de dados, cada um deles começando com um cabeçalho (rótulo em letras maiúsculas) precedido por um asterisco (*), o que permite a seleção do bloco a ser lido. Os sinais atribuídos aos elementos geométricos possibilitam definir a orientação dos componentes do modelo (contornos, faces, envelopes, volumes). Por sua vez, os *labels* permitem associar atributos físicos aos elementos geométricos e de malha (pontos ... volumes, nós...elementos) bem como referenciar dados existentes em outro arquivo da base. Maiores detalhes sobre a base de dados neutra do GOPAC podem ser obtidos em documentação específica [Gop96, Gop99].

VII.2 – ALGUNS EXEMPLOS BÁSICOS

Visando facilitar a compreensão de modelos de interesse prático, são apresentados a seguir alguns exemplos básicos envolvendo a construção de sólidos de variedade bidimensional utilizando operadores de Euler, seguidos de exemplos variados de montagem. As informações necessárias para entender o processo de construção são apresentadas na descrição de cada exemplo. Para facilitar o entendimento, cada exemplo é acompanhado de uma figura contendo modelos planares que ilustram a evolução do processo, bem como a seqüência de operações aplicadas. Também são feitas algumas considerações quanto à orientação das semi-arestas e parâmetros utilizados pelos operadores aplicados.

Deve-se ter em mente que os operadores de Euler garantem apenas a consistência topológica. A topologia sozinha representa um objeto deformável ou de borracha, que só fica rígido quando a geometria é especificada. Caso o sólido apresente inconsistências geométricas, a seqüência de construção apresentada poderá originar sólidos de variedade múltipla ou inconsistentes. Nos exemplos apresentados, as coordenadas dos vértices já foram calculadas e estão geometricamente consistentes.

VII.2.1 - A GERAÇÃO DE UM BLOCO

A geração de um bloco pode ser realizada por meio da varredura translacional simples. As faces laterais são compostas por quatro arestas, formando retângulos quando a translação for reta, ou paralelogramos quando a translação for oblíqua. Primeiro são construídas as bases e, em seguida, as faces laterais. A face superior fica automaticamente construída ao gerar a última face lateral.

Cuidado especial deve ser tomado com a orientação do sólido: todas as semi-arestas que constituem a borda de uma face (ou ciclo) devem seguir uma mesma orientação, e a normal à face, tomando por base o sentido das semi-arestas e utilizando a regra da mão direita, deve apontar para o exterior do sólido que está sendo construído.

As etapas seguidas na construção de um bloco são ilustradas na figura VII.1 e descritas no quadro VII.1. Cada etapa corresponde ao emprego de um operador de Euler e é identificada por uma seta numerada na figura. A descrição do operador empregado é obtida na linha de mesmo número existente no quadro. As setas que acompanham cada aresta na figura indicam o sentido do ciclo de semi-arestas. As setas maiores indicam que aquela semi-aresta está sendo utilizada como parâmetro para a operação seguinte. Quando um ciclo é fechado, formando uma nova face, as semi-arestas daquele ciclo não mais aparecem, por não interferirem mais nas operações restantes. Também estão anotados na figura os vértices e as faces que já foram criados.

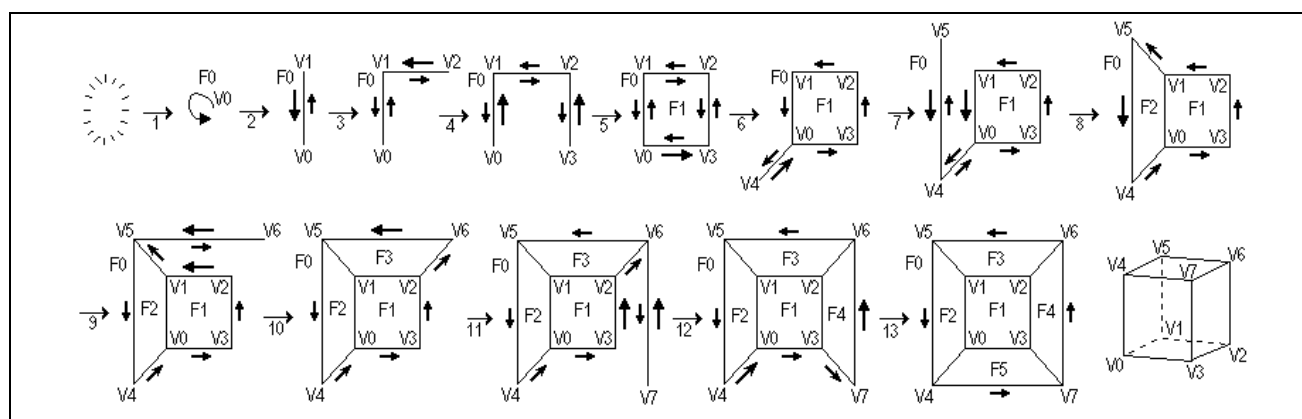



Figura VII.1 - Seqüência de operações aplicadas na construção de um bloco

01 :	MVFR	(bloco, 0, 0, 0, 0, 0.00, 0.00, 0.00)
02 :	MEV	(0, 0, 0, 0, 0, 0, 1, 0.00, 4.00, 0.00)
03 :	MEV	(0, 0, 0, 0, 1, 0, 0, 2, 4.00, 4.00, 0.00)
04 :	MEV	(0, 0, 0, 0, 2, 1, 1, 3, 4.00, 0.00, 0.00)
05 :	MEF	(0, 0, 0, 3, 2, 0, 1, 1)
06 :	MEV	(0, 0, 0, 0, 0, 3, 3, 4, 0.00, 0.00, 5.00)
07 :	MEV	(0, 0, 0, 0, 4, 0, 0, 5, 0.00, 4.00, 5.00)
08 :	MEF	(0, 0, 0, 5, 4, 1, 0, 2)
09 :	MEV	(0, 0, 0, 0, 5, 4, 4, 6, 4.00, 4.00, 5.00)
10 :	MEF	(0, 0, 0, 6, 5, 2, 1, 3)
11 :	MEV	(0, 0, 0, 0, 6, 5, 5, 7, 4.00 0.00 5.00)
12 :	MEF	(0, 0, 0, 7, 6, 3, 2, 4)
13 :	MEF	(0, 0, 0, 7, 6, 4, 0, 5)

Quadro VII.1 - Operações de Euler aplicadas na construção de um bloco

VII.2.2 - A GERAÇÃO DE SÓLIDOS OCOS

A noção de sólido contendo buracos, ou seja, oco, está intimamente relacionada à orientação das faces. Gerar uma região oca em um sólido já definido corresponde a gerar um outro sólido que ocupe o espaço oco, porém com orientação invertida, de forma que a normal a cada uma de suas faces, ao invés de apontar para fora do sólido, aponte para dentro. Esta mudança de orientação pode ser obtida invertendo a seqüência dos vértices que compõem o perfil gerador ou mantendo a seqüência e gerando os operadores de Euler no sentido oposto. Após sua construção, os dois sólidos devem ser unidos pelo operador MSKR.

A figura VII.2 ilustra a geração de um bloco com orientação invertida (interno) e a sua união com um bloco externo, gerado utilizando a seqüência apresentada para a construção de um bloco (item anterior). Note-se que o processo inicia-se no número 14, pois as 13 primeiras operações correspondem à construção do bloco externo, descrita no item anterior. O quadro VII.2 relaciona as operações aplicadas.

Figura VII.2 - Seqüência de operações aplicadas na geração de sólidos ocos

01 :	MVFR	(bloco, 0, 0, 0, 0, 0.00, 0.00, 0.00)
02 :	MEV	.(0, 0, 0, 0, 0, 0, 0, 1, 0.00, 4.00, 0.00)
03 :	MEV	.(0, 0, 0, 0, 1, 0, 0, 2, 4.00, 4.00, 0.00)
04 :	MEV	.(0, 0, 0, 0, 2, 1, 1, 3, 4.00, 0.00, 0.00)
05 :	MEF	.(0, 0, 0, 3, 2, 0, 1, 1)
06 :	MEV	.(0, 0, 0, 0, 0, 3, 3, 4, 0.00, 0.00, 5.00)
07 :	MEV	.(0, 0, 0, 0, 4, 0, 0, 5, 0.00, 4.00, 5.00)
08 :	MEF	.(0, 0, 0, 5, 4, 1, 0, 2)
09 :	MEV	.(0, 0, 0, 0, 5, 4, 4, 6, 4.00, 4.00, 5.00)
10 :	MEF	.(0, 0, 0, 6, 5, 2, 1, 3)
11 :	MEV	.(0, 0, 0, 0, 6, 5, 5, 7, 4.00 0.00 5.00)
12 :	MEF	.(0, 0, 0, 7, 6, 3, 2, 4)
13 :	MEF	.(0, 0, 0, 7, 6, 4, 0, 5)
14 :	MVFR	(bloco-interno, 1, 1, 6, 8, 1.00, 1.00, 1.00)
15 :	MEV	(1, 1, 6, 6, 8, 8, 8, 9, 1.00, 3.00, 1.00)
16 :	MEV	(1, 1, 6, 6, 9, 8, 8, 10, 3.00, 3.00, 1.00)
17 :	MEV	(1, 1, 6, 6, 10, 9, 9, 11, 3.00, 1.00, 1.00)
18 :	MEF	(1, 1, 6, 8, 9, 11, 10, 7)
19 :	MEV	(1, 1, 6, 6, 8, 9, 9, 12, 1.00, 1.00, 4.00)
20 :	MEV	(1, 1, 6, 6, 12, 8, 8, 13, 1.00, 3.00, 4.00)
21 :	MEF	(1, 1, 6, 9, 10, 13, 12, 8)
22 :	MEV	(1, 1, 6, 6, 13, 9, 9, 14, 3.00, 3.00, 4.00)
23 :	MEF	(1, 1, 6, 10, 11, 14, 13, 9)
24 :	MEV	(1, 1, 6, 6, 14, 10, 10, 15, 3.00, 1.00, 4.00)
25 :	MEF	(1, 1, 6, 11, 8, 15, 14, 10)
26 :	MEF	(1, 1, 6, 12, 13, 15, 11, 11)
27 :	MSKR	(0, 1)

Quadro VII.2 - Operações de Euler aplicadas na geração de um sólido oco

VII.2.3 - A Geração de Sólidos Vazados

A criação de vazamentos pode ser realizada paralelamente à construção ou em sólidos B-rep já definidos. Para isso, é necessário criar um anel relativo ao polígono que irá gerar o vazamento na face correspondente, por meio do operador KEML, e construir as faces laterais internas sobre esse anel, gerando uma cavidade que atinge o lado oposto do sólido e constrói automaticamente uma face interna contraposta ao anel inicial, encostada na face oposta àquela onde se iniciou o processo. A seguir, deve-

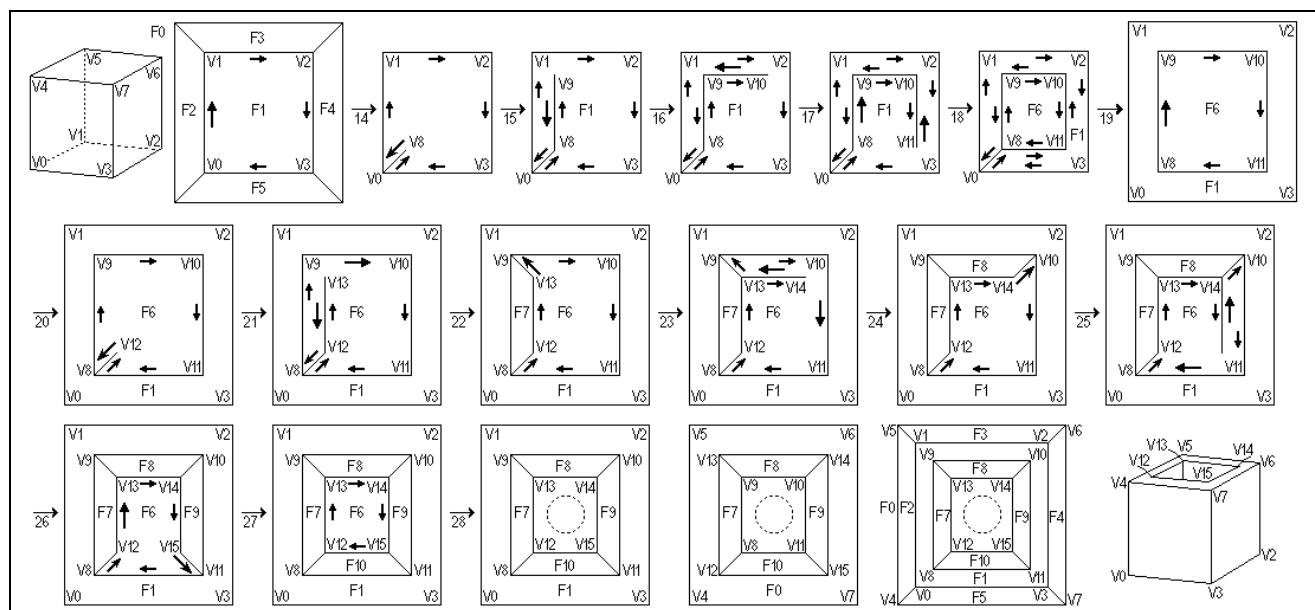



Figura VII.3 - Seqüência de operações aplicadas na geração de sólidos vazados

se aplicar o operador KFMLH para destruir essa face interna criada e fazer dela um anel na face inicial, construindo o buraco relativo ao polígono gerador. A figura VII.3 ilustra esse processo, partindo de um bloco já construído, e o quadro VII.3 descreve as operações utilizadas. É importante observar a orientação do sólido, para que o vazamento possa ser corretamente gerado.

01 :	MVFR	(blocoVazado, 0, 0, 0, 0, 0.00, 0.00, 0.00)
02 :	MEV	(0, 0, 0, 0, 0, 0, 0, 1, 0.00, 4.00, 0.00)
03 :	MEV	(0, 0, 0, 0, 1, 0, 0, 2, 4.00, 4.00, 0.00)
04 :	MEV	(0, 0, 0, 0, 2, 1, 1, 3, 4.00, 0.00, 0.00)
05 :	MEF	(0, 0, 0, 3, 2, 0, 1, 1)
06 :	MEV	(0, 0, 0, 0, 0, 3, 3, 4, 0.00, 0.00, 5.00)
07 :	MEV	(0, 0, 0, 0, 4, 0, 0, 5, 0.00, 4.00, 5.00)
08 :	MEF	(0, 0, 0, 5, 4, 1, 0, 2)
09 :	MEV	(0, 0, 0, 0, 5, 4, 4, 6, 4.00, 4.00, 5.00)
10 :	MEF	(0, 0, 0, 6, 5, 2, 1, 3)
11 :	MEV	(0, 0, 0, 0, 6, 5, 5, 7, 4.00 0.00 5.00)
12 :	MEF	(0, 0, 0, 7, 6, 3, 2, 4)
13 :	MEF	(0, 0, 0, 7, 6, 4, 0, 5)
14 :	MEV	(0, 0, 1, 1, 0, 1, 1, 8, 1.00, 1.00, 0.00)
15 :	MEV	(0, 0, 1, 1, 8, 0, 0, 9, 1.00, 3.00, 0.00)
16 :	MEV	(0, 0, 1, 1, 9, 8, 8, 10, 3.00, 3.00, 0.00)
17 :	MEV	(0, 0, 1, 1, 10, 9, 9, 11, 3.00, 1.00, 0.00)
18 :	MEF	(0, 0, 1, 11, 10, 8, 9, 6)
19 :	KEML	(0, 0, 1, 0, 8)
20 :	MEV	(0, 0, 6, 6, 8, 9, 9, 12, 1.00, 1.00, 5.00)
21 :	MEV	(0, 0, 6, 6, 12, 8, 8, 13, 1.00, 3.00, 5.00)
22 :	MEF	(0, 0, 6, 9, 10, 13, 12, 7)
23 :	MEV	(0, 0, 6, 6, 13, 9, 9, 14, 3.00, 3.00, 5.00)
24 :	MEF	(0, 0, 6, 10, 11, 14, 13, 8)
25 :	MEV	(0, 0, 6, 6, 14, 10, 10, 15, 3.00, 1.00, 5.00)
26 :	MEF	(0, 0, 6, 11, 8, 15, 14, 9)
27 :	MEF	(0, 0, 6, 15, 11, 12, 13, 10)
28 :	KFMLH	(0, 0, 10)

Quadro VII.3 - Operações de Euler aplicadas na geração de um sólido com vazamento

VII.2.4 - A COLAGEM E A MONTAGEM DE SÓLIDOS

Para que seja possível realizar uma montagem entre dois sólidos (ou duas regiões), é imprescindível que existam duas faces idênticas e coincidentes – uma em cada sólido – ocupando uma mesma posição física no espaço. Cada uma dessas faces deve possuir somente um uso-face, orientado de forma que sua normal aponte para fora do sólido definido pela face. Assim, os uso-faces das duas faces coincidentes deverão possuir orientações opostas. A operação de montagem elimina uma dessas faces e atribui seu uso-face à outra face, que se transforma em uma face de fronteira e passa a possuir dois uso-faces, cada um vinculado a um dos sólidos iniciais.

A figura VII.4 ilustra a geração de dois blocos – um maior, contendo um anel em torno da face a ser montada, outro menor, com uma face idêntica ao primeiro bloco – e a montagem destes blocos, seguindo a abordagem descrita. O quadro VII.4 relaciona as operações aplicadas. Note-se que as operações de 34 a 47 correspondem ao operador adicional *Assemble Face*, criado para simplificar o processo de construção de uma montagem.

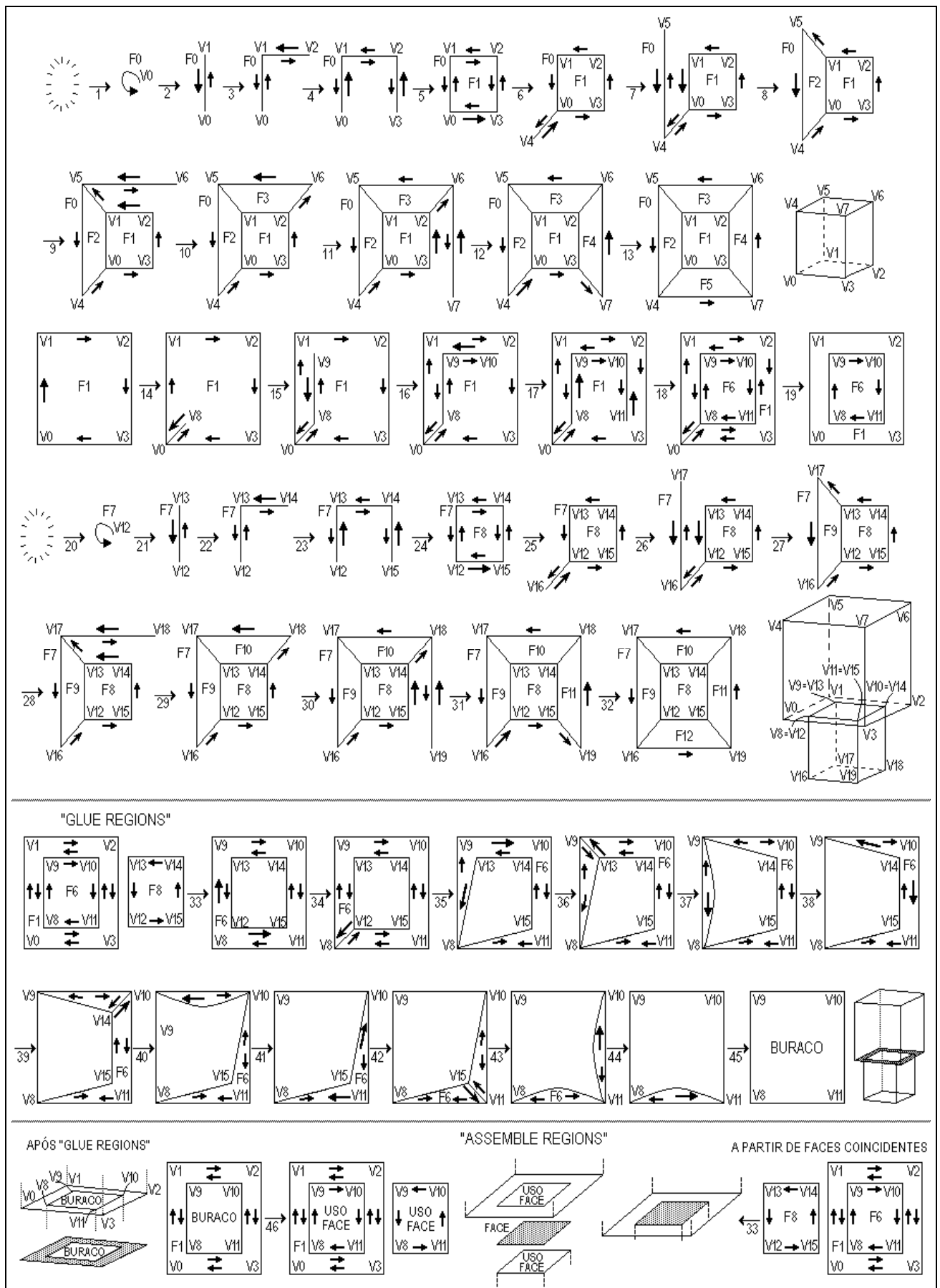



Figura VII.4 - Seqüência de operações aplicadas na união de regiões e geração de montagens

```
01 :    MVFR    (bloco1, 0, 0, 0, 0, 0.00, 0.00, 0.00)
02 :    MEV     (0, 0, 0, 0, 0, 0, 0, 1, 0.00, 4.00, 0.00)
03 :    MEV     (0, 0, 0, 0, 1, 0, 0, 2, 4.00, 4.00, 0.00)
04 :    MEV     (0, 0, 0, 0, 2, 1, 1, 3, 4.00, 0.00, 0.00)
05 :    MEF     (0, 0, 0, 3, 2, 0, 1, 1)
06 :    MEV     (0, 0, 0, 0, 0, 3, 3, 4, 0.00, 0.00, 5.00)
07 :    MEV     (0, 0, 0, 0, 4, 0, 0, 5, 0.00, 4.00, 5.00)
08 :    MEF     (0, 0, 0, 5, 4, 1, 0, 2)
09 :    MEV     (0, 0, 0, 0, 5, 4, 4, 6, 4.00, 4.00, 5.00)
10 :    MEF     (0, 0, 0, 6, 5, 2, 1, 3)
11 :    MEV     (0, 0, 0, 0, 6, 5, 5, 7, 4.00 0.00 5.00)
12 :    MEF     (0, 0, 0, 7, 6, 3, 2, 4)
13 :    MEF     (0, 0, 0, 7, 6, 4, 0, 5)
14 :    MEV     (0, 0, 1, 1, 0, 1, 1, 8, 1.00, 1.00, 0.00)
15 :    MEV     (0, 0, 1, 1, 8, 0, 0, 9, 1.00, 3.00, 0.00)
16 :    MEV     (0, 0, 1, 1, 9, 8, 8, 10, 3.00, 3.00, 0.00)
17 :    MEV     (0, 0, 1, 1, 10, 9, 9, 11, 3.00, 1.00, 0.00)
18 :    MEF     (0, 0, 1, 11, 10, 8, 9, 6)
19 :    KEMR    (0, 0, 1, 0, 8)
20 :    MVFR    (bloco2, 1, 1, 7, 12, 1.00, 1.00, -4.00)
21 :    MEV     (1, 1, 7, 7, 12, 12, 12, 13, 1.00, 3.00, -4.00)
22 :    MEV     (1, 1, 7, 7, 13, 12, 12, 14, 3.00, 3.00, -4.00)
23 :    MEV     (1, 1, 7, 2, 14, 13, 13, 15, 3.00, 1.00, -4.00)
24 :    MEF     (1, 1, 7, 15, 14, 12, 13, 8)
25 :    MEV     (1, 1, 7, 7, 12, 15, 15, 16, 1.00, 1.00, 0.00)
26 :    MEV     (1, 1, 7, 7, 16, 12, 12, 17, 1.00, 3.00, 0.00)
27 :    MEF     (1, 1, 7, 17, 16, 13, 12, 9)
28 :    MEV     (1, 1, 7, 7, 17, 16, 16, 18, 3.00, 3.00, 0.00)
29 :    MEF     (1, 1, 7, 18, 17, 14, 13, 10)
30 :    MEV     (1, 1, 7, 7, 18, 17, 17, 19, 3.00, 1.00, 0.00)
31 :    MEF     (1, 1, 7, 19, 18, 15, 14, 11)
32 :    MEF     (1, 1, 7, 19, 18, 16, 12, 12)
33 :    (mesclagem da casca 1 com a casca 0, permanecendo a 0)
34 :    KFMLH   (0, 1, 8)
35 :    MEKL    (0, 0, 6, 8, 9, 12, 15)
36 :    KEV     (0, 0, 6, 12, 8)
37 :    MEF     (0, 0, 6, 9, 10, 13, 8, 13)
38 :    KEV     (0, 0, 6, 13, 9)
39 :    KEF     (0, 0, 13, 9, 8)
40 :    MEF     (0, 0, 6, 14, 10, 14, 9, 14)
41 :    KEV     (0, 0, 6, 14, 10)
42 :    KEF     (0, 0, 14, 10, 9)
43 :    MEF     (0, 0, 6, 15, 10, 11, 8, 15)
44 :    KEV     (0, 0, 6, 15, 15, 11)
45 :    KEF     (0, 0, 15, 11, 10)
46 :    KEF     (0, 0, 6, 8, 11)
47 :    MFFR    (bloco3, 2, 0, 16)
```

Quadro VII.4 - Operações de Euler aplicadas na geração de uma montagem

VII.3 - A Geração de Modelos B-rep Simples

Os exemplos anteriores foram elaborados visando apenas à compreensão do processo de construção de sólidos B-rep utilizando os operadores de Euler, sem considerar fatores importantes para o eletromagnetismo, como o interesse pela geração de malha triangularizada sob a superfície dos modelos e a necessidade de compatibilizar a malha superficial. Apesar de ainda simples, os exemplos apresentados a seguir ilustram a geração de malha superficial utilizando o processo de varredura nas faces laterais e o programa *Triangle* para a discretização das outras faces, definidas a partir do perfil gerador.

VII.3.1 – Um Modelo em Forma de L

Este exemplo define um modelo gerado a partir de um perfil em forma de L, que poderia estar representando, por exemplo, um tubo de fluxo magnético para o qual se deseja determinar a densidade magnética. A figura VII.5 ilustra as etapas para a geração da malha sobre esse modelo e os quadros VII.5 e VII.6 relacionam a seqüência de operadores de Euler empregada na construção das faces laterais, superior e inferior do modelo.

a) varredura translacional b) malha gerada pelo *Triangle* c) modelo final

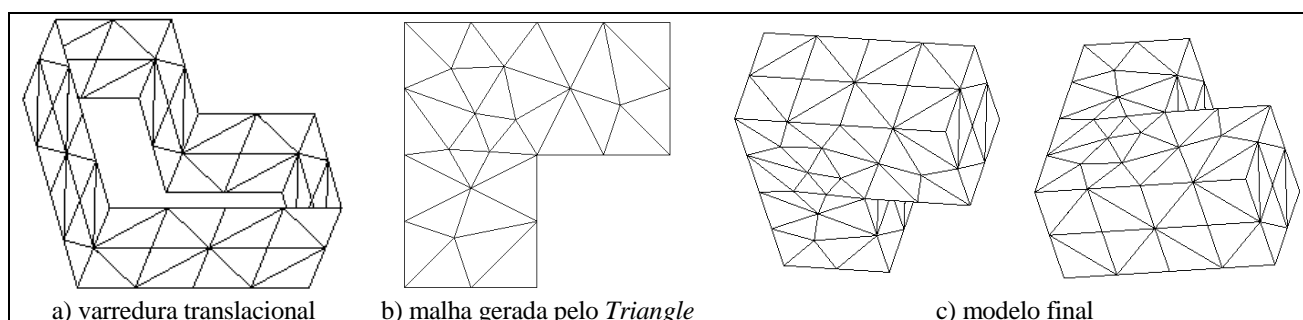



Figura VII.5 – Etapas na geração do modelo em forma de L

MVFRS	:	0 20 0 0 0	MEF	:	0 0 0 23 7 22 7	MEF	:	0 0 0 20 36 35 19
MEV	:	0 0 0 0 0 0 0 1 20.5 0 0	MEV	:	0 0 0 0 8 9 9 24 22 2 0.5	MEF	:	0 0 0 36 20 35 20
MEV	:	0 0 0 0 1 0 0 2 21 0 0	MEF	:	0 0 0 24 8 7 8	MEV	:	0 0 0 0 21 22 22 37 21.5 1 1
MEV	:	0 0 0 0 2 1 1 3 21 0.5 0	MEF	:	0 0 0 24 8 23 7	MEF	:	0 0 0 37 21 20 21
MEV	:	0 0 0 0 3 2 2 4 21 1 0	MEV	:	0 0 0 0 9 10 10 25 21.5 2 0.5	MEF	:	0 0 0 37 21 36 20
MEV	:	0 0 0 0 4 3 3 5 21.5 1 0	MEF	:	0 0 0 9 25 24 8	MEV	:	0 0 0 0 22 23 23 38 22 1 1
MEV	:	0 0 0 0 5 4 4 6 22 1 0	MEF	:	0 0 0 25 9 24 9	MEF	:	0 0 0 22 38 37 21
MEV	:	0 0 0 0 6 5 5 7 22 1.5 0	MEV	:	0 0 0 0 10 11 11 26 21 2 0.5	MEF	:	0 0 0 38 22 37 22
MEV	:	0 0 0 0 7 6 6 8 22 2 0	MEF	:	0 0 0 26 10 9 10	MEV	:	0 0 0 0 23 24 24 39 22 1.5 1
MEV	:	0 0 0 0 8 7 7 9 21.5 2 0	MEF	:	0 0 0 26 10 25 9	MEF	:	0 0 0 39 23 22 23
MEV	:	0 0 0 0 9 8 8 10 21 2 0	MEV	:	0 0 0 0 11 12 12 27 20.5 2 0.5	MEF	:	0 0 0 39 23 38 22
MEV	:	0 0 0 0 10 9 9 11 20.5 2 0	MEF	:	0 0 0 11 27 26 10	MEV	:	0 0 0 0 24 25 25 40 22 2 1
MEV	:	0 0 0 0 11 10 10 12 20 2 0	MEF	:	0 0 0 27 11 26 11	MEF	:	0 0 0 24 40 39 23
MEV	:	0 0 0 0 12 11 11 13 20 1.5 0	MEV	:	0 0 0 0 12 13 13 28 20 2 0.5	MEF	:	0 0 0 40 24 39 24
MEV	:	0 0 0 0 13 12 12 14 20 1 0	MEF	:	0 0 0 28 12 11 12	MEV	:	0 0 0 0 25 26 26 41 21.5 2 1
MEV	:	0 0 0 0 14 13 13 15 20 0.5 0	MEF	:	0 0 0 28 12 27 11	MEF	:	0 0 0 41 25 24 25
MEF	:	0 0 0 0 1 15 14	MEV	:	0 0 0 0 13 14 14 29 20 1.5 0.5	MEF	:	0 0 0 41 25 40 24
MEV	:	0 0 0 0 0 1 1 16 20 0 0.5	MEF	:	0 0 0 13 29 28 12	MEV	:	0 0 0 0 26 27 27 42 21 2 1
MEV	:	0 0 0 0 1 2 2 17 20.5 0 0.5	MEF	:	0 0 0 29 13 28 13	MEF	:	0 0 0 26 42 41 25
MEF	:	0 0 0 1 17 16 0	MEV	:	0 0 0 0 14 15 15 30 20 1 0.5	MEF	:	0 0 0 42 26 41 26
MEF	:	0 0 0 17 1 16 1	MEF	:	0 0 0 30 14 13 14	MEV	:	0 0 0 0 27 28 28 43 20.5 2 1
MEV	:	0 0 0 0 2 3 3 18 21 0 0.5	MEF	:	0 0 0 30 14 29 13	MEF	:	0 0 0 43 27 26 27
MEF	:	0 0 0 18 2 1 2	MEV	:	0 0 0 0 15 0 0 31 20 0.5 0.5	MEF	:	0 0 0 43 27 42 26
MEF	:	0 0 0 18 2 17 1	MEF	:	0 0 0 15 31 30 14	MEV	:	0 0 0 0 28 29 29 44 20 2 1
MEV	:	0 0 0 0 3 4 4 19 21 0.5 0.5	MEF	:	0 0 0 31 15 30 15	MEF	:	0 0 0 28 44 43 27
MEF	:	0 0 0 3 19 18 2	MEF	:	0 0 0 16 17 15 0	MEF	:	0 0 0 44 28 43 28
MEF	:	0 0 0 19 3 18 3	MEF	:	0 0 0 16 17 31 15	MEV	:	0 0 0 0 29 30 30 45 20 1.5 1
MEV	:	0 0 0 0 4 5 5 20 21 1 0.5	MEV	:	0 0 0 0 16 17 17 32 20 0 1	MEF	:	0 0 0 45 29 28 29
MEF	:	0 0 0 20 4 3 4	MEV	:	0 0 0 0 17 18 18 33 20.5 0 1	MEF	:	0 0 0 45 29 44 28
MEF	:	0 0 0 20 4 19 3	MEF	:	0 0 0 33 17 16 17	MEV	:	0 0 0 0 30 31 31 46 20 1 1
MEV	:	0 0 0 0 5 6 6 21 21.5 1 0.5	MEF	:	0 0 0 33 17 32 16	MEF	:	0 0 0 30 46 45 29
MEF	:	0 0 0 5 21 20 4	MEV	:	0 0 0 0 18 19 19 34 21 0 1	MEF	:	0 0 0 46 30 45 30
MEF	:	0 0 0 21 5 20 5	MEF	:	0 0 0 18 34 33 17	MEV	:	0 0 0 0 31 16 16 47 20 0.5 1
MEV	:	0 0 0 0 6 7 7 22 22 1 0.5	MEF	:	0 0 0 34 18 33 18	MEF	:	0 0 0 47 31 30 31
MEF	:	0 0 0 22 6 5 6	MEV	:	0 0 0 0 19 20 20 35 21 0.5 1	MEF	:	0 0 0 47 31 46 30
MEF	:	0 0 0 22 6 21 5	MEF	:	0 0 0 35 19 18 19	MEF	:	0 0 0 16 32 47 31
MEV	:	0 0 0 0 7 8 8 23 22 1.5 0.5	MEF	:	0 0 0 35 19 34 18	MEF	:	0 0 0 32 33 47 16
MEF	:	0 0 0 7 23 22 6	MEV	:	0 0 0 0 20 21 21 36 21 1 1			

Quadro VII.5 - Operações de Euler aplicadas na geração das faces laterais do modelo em forma de L – arquivo ".brp"

Face Inferior			Face Superior		
MEV	:	0 0 1 1 15 14 14 48 20.5 0.75 0	MEV	:	0 0 0 0 32 33 33 62 20.375 0.375 1
MEV	:	0 0 1 1 14 13 13 54 20.3331928253174 1.04138517379761 0	MEV	:	0 0 0 0 35 36 36 58 20.5 0.75 1
MEV	:	0 0 1 1 13 12 12 50 20.53125 1.3125 0	MEV	:	0 0 0 0 36 37 37 59 21.25 1.5 1
MEV	:	0 0 1 1 12 11 11 57 20.3512229919434 1.64877712726593 0	MEV	:	0 0 0 0 37 38 38 63 21.625 1.375 1
MEV	:	0 0 1 1 11 10 10 51 20.75 1.66690337657928 0	MEV	:	0 0 0 0 42 43 43 61 20.75 1.66690337657928 1
MEV	:	0 0 1 1 10 9 9 49 21.25 1.5 0	MEV	:	0 0 0 0 43 44 44 67 20.3512229919434 1.64877712726593 1
MEV	:	0 0 1 1 9 8 8 53 21.625 1.375 0	MEV	:	0 0 0 0 45 46 46 60 20.53125 1.3125 1
MEV	:	0 0 1 1 4 3 3 55 20.6412048339844 1.02427327632904 0	MEV	:	0 0 0 0 46 47 47 64 20.3331928253174 1.04138517379761 1
MEV	:	0 0 1 1 3 2 2 52 20.375 0.375 0	MEV	:	0 0 0 0 58 35 35 65 20.6412048339844 1.02427327632904 1
MEV	:	0 0 1 1 50 13 13 56 20.8376636505127 1.26430654525757 0	MEV	:	0 0 0 0 59 36 36 66 20.8376636505127 1.26430654525757 1
MEF	:	0 0 1 14 54 48 15	MEF	:	0 0 0 33 34 62 32
MEF	:	0 0 1 54 14 48 14	MEF	:	0 0 0 35 58 33 34
MEF	:	0 0 1 13 50 54 14	MEF	:	0 0 0 35 58 62 33
MEF	:	0 0 1 50 56 54 13	MEF	:	0 0 0 58 65 62 35
MEF	:	0 0 1 57 12 13 12	MEF	:	0 0 0 36 59 58 35
MEF	:	0 0 1 57 12 50 13	MEF	:	0 0 0 36 59 65 58
MEF	:	0 0 1 11 51 57 12	MEF	:	0 0 0 66 59 36 59
MEF	:	0 0 1 51 11 57 11	MEF	:	0 0 0 66 59 65 36
MEF	:	0 0 1 51 11 50 57	MEF	:	0 0 0 37 63 59 36
MEF	:	0 0 1 51 11 56 50	MEF	:	0 0 0 63 37 59 37
MEF	:	0 0 1 10 49 51 11	MEF	:	0 0 0 38 39 63 37
MEF	:	0 0 1 49 10 51 10	MEF	:	0 0 0 39 40 63 38
MEF	:	0 0 1 49 10 56 51	MEF	:	0 0 0 41 42 39 40
MEF	:	0 0 1 9 53 49 10	MEF	:	0 0 0 41 42 63 39
MEF	:	0 0 1 53 9 49 9	MEF	:	0 0 0 41 42 59 63
MEF	:	0 0 1 7 6 9 8	MEF	:	0 0 0 42 61 59 41
MEF	:	0 0 1 7 6 53 9	MEF	:	0 0 0 61 42 59 42
MEF	:	0 0 1 6 5 53 7	MEF	:	0 0 0 61 42 66 59
MEF	:	0 0 1 5 4 53 6	MEF	:	0 0 0 43 67 61 42
MEF	:	0 0 1 5 4 49 53	MEF	:	0 0 0 67 43 61 43
MEF	:	0 0 1 4 55 49 5	MEF	:	0 0 0 44 45 67 43
MEF	:	0 0 1 4 55 56 49	MEF	:	0 0 0 45 60 67 44
MEF	:	0 0 1 55 4 56 4	MEF	:	0 0 0 60 45 67 45
MEF	:	0 0 1 55 4 50 56	MEF	:	0 0 0 60 45 61 67
MEF	:	0 0 1 55 4 54 50	MEF	:	0 0 0 60 45 66 61
MEF	:	0 0 1 55 4 48 54	MEF	:	0 0 0 60 45 65 66
MEF	:	0 0 1 4 3 48 55	MEF	:	0 0 0 64 46 45 46
MEF	:	0 0 1 3 52 48 4	MEF	:	0 0 0 64 46 60 45
MEF	:	0 0 1 52 3 48 3	MEF	:	0 0 0 64 46 65 60
MEF	:	0 0 1 52 3 15 48	MEF	:	0 0 0 64 46 58 65
MEF	:	0 0 1 1 0 3 2	MEF	:	0 0 0 46 47 58 64
MEF	:	0 0 1 1 0 52 3	MEF	:	0 0 0 47 32 58 46
MEF	:	0 0 1 0 15 52 1	MEF	:	0 0 0 47 32 62 58

Quadro VII.6 - Operações de Euler aplicadas na geração das faces inferior e superior do modelo em forma de L – arq ".brp"

Os quadros VII.7 e VII.8 apresentam os arquivos neutros de geometria e malha correspondentes. Nota-se que, para exemplos ainda muito simples, tanto o arquivo contendo os operadores de Euler para a construção do sólido quanto os que compõem a base de dados neutra possuem tamanho significativo, que tende a crescer muito à medida que o modelo aumenta em complexidade e quantidade de elementos compondo a malha superficial resultante. Esse mesmo perfil pode ser utilizado com diversos refinamentos de malha bem como em outras formas de varredura, como ilustra a figura VII.6. O quadro VII.9 mostra o arquivo ".3DF" para visualização estéreo do modelo.

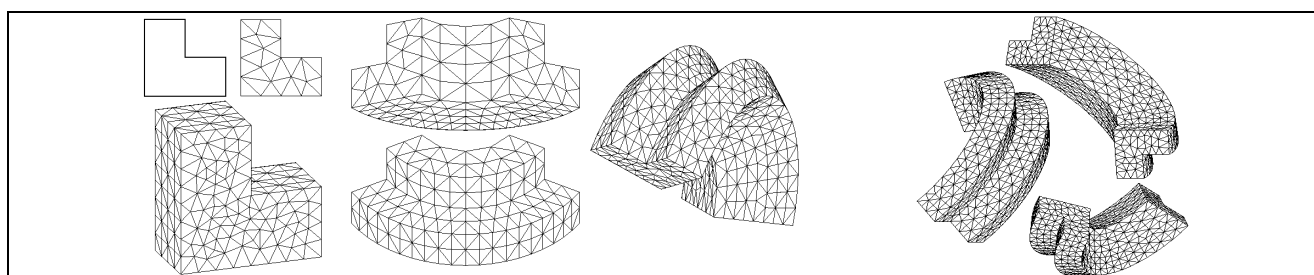



Figura VII.6 – Variações de modelos gerados a partir do perfil em forma de L

```
REMARK
teste
*HEADER.FILE
teste
*HEADER.AUTHOR
teste
*HEADER.DATE
22/2/2000
*HEADER.VERSION
3.1
*HEADER.TITLE
teste
*HEADER.ANALYSIS
Geração de Malha
*HEADER.UNITS
M KG SEG DEG
*HEADER.DIM
3_D
*POINT 68 *POINT.COORD
1 20 0 0          10 21.5 2 0 11   19 21 0 0.5      28 20.5 2 0.5    37 21 1 1 38     46 20 1.5 1 47 20 1         55 21.625 1.375 0     64 20.3512 1.6487 1
2 20.5 0 0        21 2 0 12 20.5   20 21 0.5 0.5    29 20 2 0.5      21.5 1 1 39 22   1 48 20 0.5 1 49 20.5       56 20.6412 1.0242 0   65 20.5312 1.3125 1
3 21 0 0          2 0 13 20 2      21 21 1 0.5      30 20 1.5 0.5    1 1 40 22 1.5    0.75 0 50 20.3332           57 20.375 0.375 0     66 20.3332 1.0414 1
4 21 0.5 0        0 14 20 1.5      22 21.5 1 0.5    31 20 1 0.5      1 41 22 2 1 42   1.0414 0 51 20.5312         58 20.8376 1.2643 0   67 20.6412 1.0243 1
5 21 1 0          0 15 20 1 0 16   23 22 1 0.5      32 20 0.5 0.5    21.5 2 1 43 21   1.3125 0 52 20.3512         59 20.375 0.375 1     68 20.8376 1.2643 1
6 21.5 1 0        20 0.5 0 17 20   24 22 1.5 0.5    33 20 0 1        2 1 44 20.5 2    1.6487 0 53 20.75           60 20.5 0.75 1
7 22 1 0 8 22     0 0.5 18 20.5    25 22 2 0.5      34 20.5 0 1      1 45 20 2 1      1.6669 0 54 21.25 1.5 0     61 21.25 1.5 1
1.5 0 9 22 2 0    0 0.5            26 21.5 2 0.5    35 21 0 1                                                     62 21.625 1.375 1 63
                                   27 21 2 0.5      36 21 0.5 1                                                   20.75 1.6669 1
*LINE 198 *LINE.SEGMENT.CONTENTS 198
1 2 1 2       21 2 3 19     41 2 24 25 42   61 2 16 31 62   81 2 23 39    101 2 44 45   121 2 4 57    141 2 6 55    161 2 44 64   181 2 43 61
2 2 2 3       22 2 2 19     2  10  26 43  2 2  31  32 63  2 82 2 23 38    102 2 30 46   122 2 51 58   142 2 6 54    162 2 46 65   182 2 61 63
3 2 3 4       23 2 18 19    10 25 44 2 25   16 17 64 2 17   83 2 38 39    103 2 29 46   123 2 15 49   143 2 5 54    163 2 47 66   183 2 63 68
4 2 4 5       24 2 4 20     26 45   2   11  32 65   2   17  84 2 24 40    104 2 45 46   124 2 49 50   144 2 5 58    164 2 60 67   184 2 44 63
5 2 5 6       25 2 4 19     27 46   2   10  33 66   2   18  85 2 23 40    105 2 31 47   125 2 14 50   145 2 56 58   165 2 61 68   185 2 63 64
6 2 6 7       26 2 19 20    27 47   2   26  34 67   2   17  86 2 39 40    106 2 31 46   126 2 50 51   146 2 51 56   166 2 34 59   186 2 45 64
7 2 7 8       27 2 5 21     27 48   2   12  34 68   2   33  87 2 25 41    107 2 46 47   127 2 14 52   147 2 50 56   167 2 34 36   187 2 46 64
8 2 8 9       28 2 4 21     28 49   2   12  34 69   2   19  88 2 25 40    108 2 32 48   128 2 51 52   148 2 49 56   168 2 36 59   188 2 64 65
9 2 9 10      29 2 20 21    27 50   2   27  35 70   2   19  89 2 40 41    109 2 31 48   129 2 12 52   149 2 5 49    169 2 59 60   189 2 63 65
10 2 10 11    30 2 6 22     28 51   2   13  34 71   2   34  90 2 26 42    110 2 47 48   130 2 52 53   150 2 4 49    170 2 37 60   190 2 65 68
11 2 11 12    31 2 6 21     29 52   2   12  35 72   2   20  91 2 25 42    111 2 17 48   131 2 51 53   151 2 49 57   171 2 37 67   191 2 65 67
12 2 12 13    32 2 21 22    29 53   2   28  36 73   2   19  92 2 41 42    112 2 33 48   132 2 53 58   152 2 16 57   172 2 37 68   192 2 46 66
13 2 13 14    33 2 7 23     29 54   2   14  36 74   2   35  93 2 27 43    113 2 16 49   133 2 11 53   153 2 2 4     173 2 67 68   193 2 65 66
14 2 14 15    34 2 6 23     30 55   2   14  36 75   2   21  94 2 27 42    114 2 15 50   134 2 53 54   154 2 2 57    174 2 38 61   194 2 66 67
15 2 15 16    35 2 22 23    29 56   2   29  37 76   2   21  95 2 42 43    115 2 14 51   135 2 54 58   155 2 1 57    175 2 61 62   195 2 60 66
16 2 1 16     36 2 8 24     30 57   2   15  36 77   2   36  96 2 28 44    116 2 13 52   136 2 10 54   156 2 33 59   176 2 39 62   196 2 47 60
17 2 1 17     37 2 8 23     31 58   2   14  37 78   2   22  97 2 27 44    117 2 12 53   137 2 54 55   157 2 36 60   177 2 40 62   197 2 48 60
18 2 2 18     38 2 23 24    31 59   2   30  38 79   2   21  98 2 43 44    118 2 11 54   138 2 8 10    158 2 37 61   178 2 40 42   198 2 48 59
19 2 2 17     39 2 9 25     31 60 2 16 32   38 80 2 37 38   99 2 29 45    119 2 10 55   139 2 8 55    159 2 38 62   179 2 42 62
20 2 17 18    40 2 8 25                                     100 2 29 44   120 2 5 56    140 2 7 55    160 2 43 63   180 2 42 61
*CONTOUR 132 *CONTOUR.CONTENTS
1 3 156 -198 -112   20 3 -44 -43 42   39 3 -73 26 72    58 3 -101 -100 99    77 3 133 -117 -11    96 3 -152 113 151    115 3 181 -180 95
2 3 16 152 -155     21 3 -46 10 45    40 3 -74 -69 73   59 3 -103 56 102     78 3 134 -133 118    97 3 153 -3 -2       116 3 -182 -181 160
3 3 19 -17 1        22 3 -47 -42 46   41 3 76 -72 29    60 3 -104 -99 103    79 3 135 -132 -134   98 3 154 -121 -153   117 3 183 -165 182
4 3 -20 -19 18      23 3 49 -45 11    42 3 -77 -76 75   61 3 106 -102 59     80 3 136 -118 -10    99 3 155 -154 -1     118 3 184 -160 98
5 3 -22 2 21        24 3 -50 -49 48   43 3 -79 32 78    62 3 -107 -106 105   81 3 -137 -136 119   100 3 166 -156 68    119 3 -185 -184 161
6 3 -23 -18 22      25 3 -52 12 51    44 3 -80 -75 79   63 3 -109 62 108     82 3 138 -9 -8       101 3 -167 71 74     120 3 186 -161 101
7 3 25 -21 3        26 3 -53 -48 52   45 3 82 -78 35    64 3 -110 -105 109   83 3 139 -119 -138   102 3 168 -166 167   121 3 187 -186 104
8 3 -26 -25 24      27 3 55 -51 13    46 3 -83 -82 81   65 3 111 -108 -64    84 3 140 -139 -7     103 3 169 -168 157   122 3 188 -187 162
9 3 -28 4 27        28 3 -56 -55 54   47 3 -85 38 84    66 3 112 -111 65     85 3 141 -140 -6     104 3 170 -157 77    123 3 189 185 -188
10 3 -29 -24 28     29 3 -58 14 57    48 3 -86 -81 85   67 3 123 -113 -15    86 3 142 137 -141    105 3 171 -164 -170  124 3 190 -183 -189
11 3 31 -27 5       30 3 -59 -54 58   49 3 88 -84 41    68 3 -124 -123 114   87 3 143 -142 -5     106 3 -172 158 165   125 3 191 173 -190
12 3 -32 -31 30     31 3 61 -57 15    50 3 -89 -88 87   69 3 125 -114 -14    88 3 144 -135 -143   107 3 -173 -171 172  126 3 -192 107 163
13 3 -34 6 33       32 3 -62 -61 60   51 3 -91 44 90    70 3 126 -125 115    89 3 145 -144 120    108 3 174 -158 80    127 3 -193 -162 192
14 3 -35 -30 34     33 3 -63 -16 17   52 3 -92 -87 91   71 3 -127 -13 116    90 3 -146 122 -145   109 3 -175 -174 159  128 3 194 -191 193
15 3 37 -33 7       34 3 64 -60 63    53 3 94 -90 47    72 3 -128 -115 127   91 3 -147 -126 146   110 3 176 -159 83    129 3 -195 164 -194
16 3 -38 -37 36     35 3 -67 20 66    54 3 -95 -94 93   73 3 129 -116 -12    92 3 -148 124 147    111 3 177 -176 86    130 3 196 195 -163
17 3 -40 8 39       36 3 -68 -65 67   55 3 -97 50 96    74 3 130 -129 117    93 3 149 148 -120    112 3 -178 89 92     131 3 197 -196 110
18 3 -41 -36 40     37 3 70 -66 23    56 3 -98 -93 97   75 3 -131 128 -130   94 3 150 -149 -4     113 3 179 -177 178   132 3 198 -169 –197
19 3 43 -39 9       38 3 -71 -70 69   57 3 100 -96 53   76 3 132 -122 131    95 3 -151 -150 121   114 3 180 175 -179
```

Quadro VII.7 – Arquivo neutro de geometria gerado para o modelo em forma de L (início)

```
*FACE 132 *FACE.PLANE.CONTENTS
```

```
1 1 1 2 1 2 3
1 3 4 1 4 5 1
5 6 1 6 7 1
7 8 1 8 9 1
9 10 1 10 11
1 11 12 1 12
13 1 13
14 1 14
15 1 15
16 1 16
17 1 17
18 1 18
19 1 19
20 1 20
21 1 21
22 1 22
23 1 23
24 1 24
25 1 25 26 1
26 27 1
27 28 1
28 29 1
29 30 1
30 31 1
31 32 1
32 33 1
33 34 1
34 35 1
35 36 1 36
37 1 37 38 1
38 39 1
39 40 1
40 41 1
41 42 1
42 43 1
43 44 1
44 45 1
45 46 1
46 47 1
47 48 1 48
49 1 49
50 1 50
51 1 51
52 1 52
53 1 53
54 1 54
55 1 55
56 1 56
57 1 57
58 1 58
59 1 59
60 1 60
61 1 61 62 1
62 63 1
63 64 1
64 65 1
65 66 1
66 67 1
67 68 1
68 69 1
69 70 1
70 71 1
71 72 1 72
73 1 73
74 1 74
75 1 75
76 1 76
77 1 77
78 1 78
79 1 79
80 1 80
81 1 81
82 1 82
83 1 83
84 1 84
85 1 85 86 1
86 87 1
87 88 1
88 89 1
89 90 1
90 91 1
91 92 1
92 93 1
93 94 1
94 95 1
95 96 1 96
97 1 97
98 1 98
99 1 99
100 1 100
101 1 101
102 1 102
103 1 103
104 1 104
105 1 105
106 1 106
107 1 107
108 1 108
109 1
109 110 1
110 111 1
111 112 1
112 113 1
113 114 1
114 115 1
115 116 1
116 117 1
117 118 1
118 119 1
119 120 1
120
121 1 121
122 1 122
123 1 123
124 1 124
125 1 125
126 1 126
127 1 127
128 1 128
129 1 129
130 1 130
131 1 131
132 1 132
*ENVELOPE
1
*ENVELOPE.CONTENTS
1 132 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103
104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
*VOLUME
1
*VOLUME.CONTENTS
1 1 1
*END
```

Quadro VII.7 – Arquivo neutro de geometria gerado para o modelo em forma de L (continuação)

```
*REMARK
teste
*HEADER.FILE
teste
*HEADER.AUTHOR
teste
*HEADER.DATE
22/2/2000
*HEADER.VERSION
3.1
*HEADER.TITLE
teste
*HEADER.ANALYSIS
Geração de Malha
*HEADER.UNITS
M KG SEG DEG
*HEADER.DIM
3_D
*NODE
68
*NODE.COORD
1 20 0 0
2 20.5 0 0
3 21 0 0
4 21 0.5 0
5 21 1 0
6 21.5 1 0
7 22 1 0 8 22
1.5 0 9 22 2 0
10 21.5 2 0
11 21 2 0
12 20.5 2 0
13 20 2 0
14 20 1.5 0
15 20 1 0
16 20 0.5 0
17 20 0 0.5 18
20.5 0 0.5
19 21 0 0.5
20 21 0.5 0.5
21 21 1 0.5
22 21.5 1 0.5
23 22 1 0.5
24 22 1.5 0.5
25 22 2 0.5
26 21.5 2 0.5
27 21 2 0.5
28 20.5 2 0.5
29 20 2 0.5
30 20 1.5 0.5
31 20 1 0.5
32 20 0.5 0.5
33 20 0 1
34 20.5 0 1
35 21 0 1
36 21 0.5 1
37 21 1 1 38
21.5 1 1 39 22 1
1 40 22 1.5 1 41
22 2 1 42 21.5 2
1 43 21 2 1 44
20.5 2 1 45 20 2
1
46 20 1.5 1
47 20 1 1
48 20 0.5 1
49 20.5 0.75 0
50 20.3332 1.0414 0
51 20.5312 1.3125 0
52 20.3512 1.6487 0
53 20.75 1.6669 0
54 21.25 1.5 0
55 21.625 1.375 0
56 20.6412 1.0242 0
57 20.375 0.375 0
58 20.8376 1.2643 0
59 20.375 0.375 1
60 20.5 0.75 1
61 21.25 1.5 1
62 21.625 1.375 1 63
20.75 1.6669 1
64 20.3512 1.6487 1
65 20.5312 1.3125 1
66 20.3332 1.0414 1
67 20.6412 1.0243 1
68 20.8376 1.2643 1
*ELEMENT 132 *ELEMENT.TRIANGLE3 132
1 33 59 48 T
2 1 16 57 T
3 2 17 1 T
4 18 17 2 T
5 19 2 3 T
6 19 18 2 T
7 4 19 3 T
8 20 19 4 T
9 21 4 5 T
10 21 20 4 T
11 6 21 5 T
12 22 21 6 T
13 23 6 7 T
14 23 22 6 T
15 8 23 7 T
16 24 23 8 T
17 25 8 9 T
18 25 24 8 T
19 10 25 9 T
20 26 25 10 T
21 27 10 11 T
22 27 26 10 T
23 12 27 11 T
24 28 27 12 T
25 29 12 13 T
26 29 28 12 T
27 14 29 13 T
28 30 29 14 T
29 31 14 15 T
30 31 30 14 T
31 16 31 15 T
32 32 31 16 T
33 17 16 1 T
34 17 32 16 T
35 34 17 18 T
36 34 33 17 T
37 19 34 18 T
38 35 34 19 T
39 36 19 20 T
40 36 35 19 T
41 21 36 20 T
42 37 36 21 T
43 38 21 22 T
44 38 37 21 T
45 23 38 22 T
46 39 38 23 T
47 40 23 24 T
48 40 39 23 T
49 25 40 24 T
50 41 40 25 T
51 42 25 26 T
52 42 41 25 T
53 27 42 26 T
54 43 42 27 T
55 44 27 28 T
56 44 43 27 T
57 29 44 28 T
58 45 44 29 T
59 46 29 30 T
60 46 45 29 T
61 31 46 30 T
62 47 46 31 T
63 48 31 32 T
64 48 47 31 T
65 17 48 32 T
66 33 48 17 T
67 15 49 16 T
68 50 49 15 T
69 14 50 15 T
70 51 50 14 T
71 52 14 13 T
72 52 51 14 T
73 12 52 13 T
74 53 52 12 T
75 53 51 52 T
76 53 58 51 T
77 11 53 12 T
78 54 53 11 T
79 54 58 53 T
80 10 54 11 T
81 55 54 10 T
82 8 10 9 T
83 8 55 10 T
84 7 55 8 T
85 6 55 7 T
86 6 54 55 T
87 5 54 6 T
88 5 58 54 T
89 56 58 5 T
90 56 51 58 T
91 56 50 51 T
92 56 49 50 T
93 5 49 56 T
94 4 49 5 T
95 57 49 4 T
96 57 16 49 T
97 2 4 3 T
98 2 57 4 T
99 1 57 2 T
100 34 59 33 T
101 36 34 35 T
102 36 59 34 T
103 60 59 36 T
104 37 60 36 T
105 37 67 60 T
106 68 37 61 T
107 68 67 37 T
108 38 61 37 T
109 62 61 38 T
110 39 62 38 T
111 40 62 39 T
112 42 40 41 T
113 42 62 40 T
114 42 61 62 T
115 43 61 42 T
116 63 61 43 T
117 63 68 61 T
118 44 63 43 T
119 64 63 44 T
120 45 64 44 T
121 46 64 45 T
122 65 64 46 T
123 65 63 64 T
124 65 68 63 T
125 65 67 68 T
126 66 46 47 T
127 66 65 46 T
128 66 67 65 T
129 66 60 67 T
130 47 60 66 T
131 48 60 47 T
132 48 59 60 T
*END
```

Quadro VII.8 – Arquivo neutro de malha gerado para o modelo em forma de L

```
20 0 1
20.375 0.375 1
20 0.5 1
21.5 2 0.5
22 2 0.5
21.5 2 0
21 0.5 1
21 0 0.5
21 0.5 0.5
20 2 1
20.5 2 1
20 2 0.5
21 2 0
20.75 1.6669 0
20.5 2 0
20.375 0.375 0
20 0.5 0
20.5 0.75 0
21 2 1
21.25 1.5 1
21.5 2 1
```

20 0 0	21 2 0.5	21 0.5 1	20 1.5 1	21.25 1.5 0	20.5 0 0	20.75 1.6669 1
20 0.5 0	21.5 2 0	21 0 1	20 2 0.5	20.75 1.6669 0	21 0.5 0	21.25 1.5 1
20.375 0.375 0	21 2 0	21 0 0.5	20 1.5 0.5	21 2 0	21 0 0	21 2 1
20.5 0 0	21 2 0.5	21 1 0.5	20 1.5 1	21.25 1.5 0	20.5 0 0	20.75 1.6669 1
20 0 0.5	21.5 2 0.5	21 0.5 1	20 2 1	20.8377 1.26431 0	20.375 0.375 0	20.8377 1.26431 1
20 0 0	21.5 2 0	21 0.5 0.5	20 2 0.5	20.75 1.6669 0	21 0.5 0	21.25 1.5 1
20.5 0 0.5	20.5 2 0	21 1 1	20 1 0.5	21.5 2 0	20 0 0	20.5 2 1
20 0 0.5	21 2 0.5	21 0.5 1	20 1.5 1	21.25 1.5 0	20.375 0.375 0	20.75 1.6669 1
20.5 0 0	21 2 0	21 1 0.5	20 1.5 0.5	21 2 0	20.5 0 0	21 2 1
21 0 0.5	20.5 2 0.5	21.5 1 1	20 1 1	21.625 1.375 0	20.5 0 1	20.3512 1.64878 1
20.5 0 0	21 2 0.5	21 1 0.5	20 1.5 1	21.25 1.5 0	20.375 0.375 1	20.75 1.6669 1
21 0 0	20.5 2 0	21.5 1 0.5	20 1 0.5	21.5 2 0	20 0 1	20.5 2 1
21 0 0.5	20 2 0.5	21.5 1 1	20 0.5 1	22 1.5 0	21 0.5 1	20 2 1
20.5 0 0.5	20.5 2 0	21 1 1	20 1 0.5	21.5 2 0	20.5 0 1	20.3512 1.64878 1
20.5 0 0	20 2 0	21 1 0.5	20 0.5 0.5	22 2 0	21 0 1	20.5 2 1
21 0.5 0	20 2 0.5	22 1 0.5	20 0.5 1	22 1.5 0	21 0.5 1	20 1.5 1
21 0 0.5	20.5 2 0.5	21.5 1 1	20 1 1	21.625 1.375 0	20.375 0.375 1	20.3512 1.64878 1
21 0 0	20.5 2 0	21.5 1 0.5	20 1 0.5	21.5 2 0	20.5 0 1	20 2 1
21 0.5 0.5	20 1.5 0	22 1 1	20 0 0.5	22 1 0	20.5 0.75 1	20.5312 1.3125 1
21 0 0.5	20 2 0.5	21.5 1 1	20 0.5 1	21.625 1.375 0	20.375 0.375 1	20.3512 1.64878 1
21 0.5 0	20 2 0	22 1 0.5	20 0.5 0.5	22 1.5 0	21 0.5 1	20 1.5 1
21 1 0.5	20 1.5 0.5	22 1.5 1	20 0 1	21.5 1 0	21 1 1	20.5312 1.3125 1
21 0.5 0	20 2 0.5	22 1 0.5	20 0.5 1	21.625 1.375 0	20.5 0.75 1	20.75 1.6669 1
21 1 0	20 1.5 0	22 1.5 0.5	20 0 0.5	22 1 0	21 0.5 1	20.3512 1.64878 1
21 1 0.5	20 1 0.5	22 1.5 1	20 1 0	21.5 1 0	21 1 1	20.5312 1.3125 1
21 0.5 0.5	20 1.5 0	22 1 1	20.5 0.75 0	21.25 1.5 0	20.6412 1.02427 1	20.8377 1.26431 1
21 0.5 0	20 1 0	22 1 0.5	20 0.5 0	21.625 1.375 0	20.5 0.75 1	20.75 1.6669 1
21.5 1 0	20 1 0.5	22 2 0.5	20.3332 1.04139 0	21 1 0	20.8377 1.26431 1	20.5312 1.3125 1
21 1 0.5	20 1.5 0.5	22 1.5 1	20.5 0.75 0	21.25 1.5 0	21 1 1	20.6412 1.02427 1
21 1 0	20 1.5 0	22 1.5 0.5	20 1 0	21.5 1 0	21.25 1.5 1	20.8377 1.26431 1
21.5 1 0.5	20 0.5 0	22 2 1	20 1.5 0	21 1 0	20.8377 1.26431 1	20.3332 1.04139 1
21 1 0.5	20 1 0.5	22 1.5 1	20.3332 1.04139 0	20.8377 1.26431 0	20.6412 1.02427 1	20 1.5 1
21.5 1 0	20 1 0	22 2 0.5	20 1 0	21.25 1.5 0	21 1 1	20 1 1
22 1 0.5	20 0.5 0.5	21.5 2 1	20.5312 1.3125 0	20.6412 1.02427 0	21.5 1 1	20.3332 1.04139 1
21.5 1 0	20 1 0.5	22 2 0.5	20.3332 1.04139 0	20.8377 1.26431 0	21.25 1.5 1	20.5312 1.3125 1
22 1 0	20 0.5 0	21.5 2 0.5	20 1.5 0	21 1 0	21 1 1	20 1.5 1
22 1 0.5	20 0 0.5	21.5 2 1	20.3512 1.64878 0	20.6412 1.02427 0	21.625 1.375 1	20.3332 1.04139 1
21.5 1 0.5	20 0.5 0	22 2 1	20 1.5 0	20.5312 1.3125 0	21.25 1.5 1	20.6412 1.02427 1
21.5 1 0	20 0 0	22 2 0.5	20 2 0	20.8377 1.26431 0	21.5 1 1	20.5312 1.3125 1
22 1.5 0	20 0 0.5	21 2 0.5	20.3512 1.64878 0	20.6412 1.02427 0	22 1 1	20.3332 1.04139 1
22 1 0.5	20 0.5 0.5	21.5 2 1	20.5312 1.3125 0	20.3332 1.04139 0	21.625 1.375 1	20.5 0.75 1
22 1 0	20 0.5 0	21.5 2 0.5	20 1.5 0	20.5312 1.3125 0	21.5 1 1	20.6412 1.02427 1
22 1.5 0.5	20.5 0 1	21 2 1	20.5 2 0	20.6412 1.02427 0	22 1.5 1	20 1 1
22 1 0.5	20 0 0.5	21.5 2 1	20.3512 1.64878 0	20.5 0.75 0	21.625 1.375 1	20.5 0.75 1
22 1.5 0	20.5 0 0.5	21 2 0.5	20 2 0	20.3332 1.04139 0	22 1 1	20.3332 1.04139 1
22 2 0.5	20.5 0 1	20.5 2 1	20.75 1.6669 0	21 1 0	21.5 2 1	20 0.5 1
22 1.5 0	20 0 1	21 2 0.5	20.3512 1.64878 0	20.5 0.75 0	22 1.5 1	20.5 0.75 1
22 2 0	20 0 0.5	20.5 2 0.5	20.5 2 0	20.6412 1.02427 0	22 2 1	20 1 1
22 2 0.5	21 0 0.5	20.5 2 1	20.75 1.6669 0	21 0.5 0	21.5 2 1	20 0.5 1 20.375 0.375 1 20.5 0.75 1
22 1.5 0.5	20.5 0 1	21 2 1	20.5312 1.3125 0	20.5 0.75 0	21.625 1.375 1	
22 1.5 0	20.5 0 0.5	21 2 0.5	20.3512 1.64878 0	21 1 0	22 1.5 1	
21.5 2 0 22 2 0.5 22 2 0	21 0 1 20.5 0 1 21 0 0.5	20 2 0.5 20.5 2 1 20.5 2 0.5	20.75 1.6669 0	20.375 0.375 0 20.5 0.75 0 21 0.5 0	21.5 2 1 21.25 1.5 1 21.625 1.375 1	
			20.8377 1.26431 0			
			20.5312 1.3125 0			

Quadro VII.9 – Arquivo ".3DF" para visualização estéreo e/ou das faces do modelo em L

VII.3.2 – VARIAÇÕES NA DENSIDADE DA MALHA

A densidade da malha resultante depende não só do tamanho do "segmento padrão" definido para o modelo, obtido pela média dos segmentos que definem o perfil gerador – conforme explicado nos itens VI.3.1 e VI.3.2 – mas também do número de divisões estabelecido sobre esse padrão, que pode ser alterado conforme o interesse do usuário. A figura VII.7 mostra um núcleo magnético definido por um mesmo perfil, sobre o qual se fez variar o número de divisões do segmento padrão. O mesmo caso pode ser observado na figura VII.8, que também apresenta variações na forma de varredura do perfil.

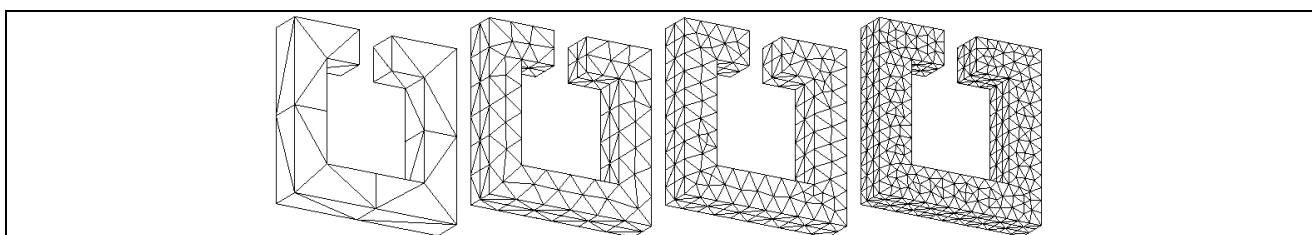



Figura VII.7 – Variações na densidade de malha devido à divisão do segmento padrão em mais elementos

a) variações utilizando um perfil em forma de U b) variações utilizando um perfil em forma de M

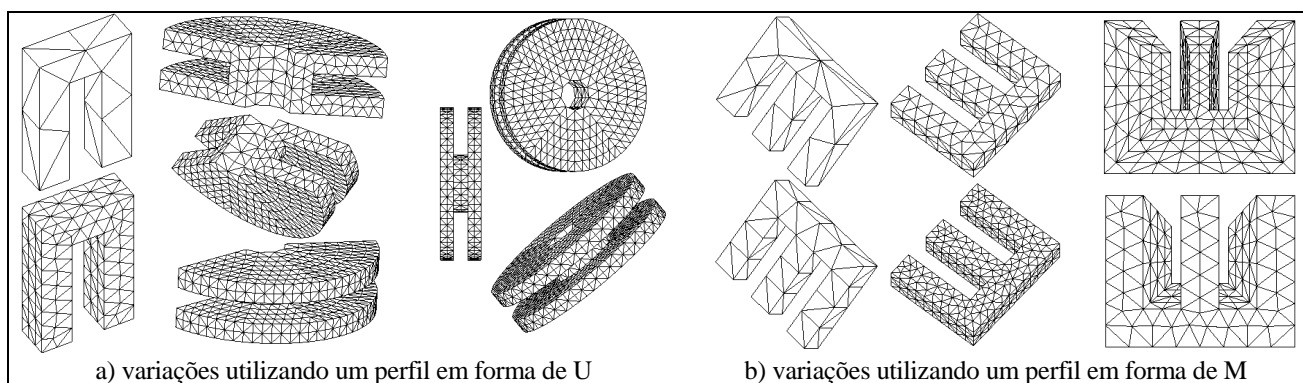



Figura VII.8 – Mais exemplos de variações na densidade de malha e na forma da varredura

Em perfis compostos, a variação na densidade da malha poderá ser manipulada conforme o interesse do usuário. A figura VII.9 mostra variações na malha superficial devido à variação no número de segmentos definido sobre cada perfil. De toda forma, a malha superficial lateral, definida pela varredura, será sempre compatível com a malha gerada sobre o perfil.

a) exemplos de variações na densidade da malha b) resultado após a varredura

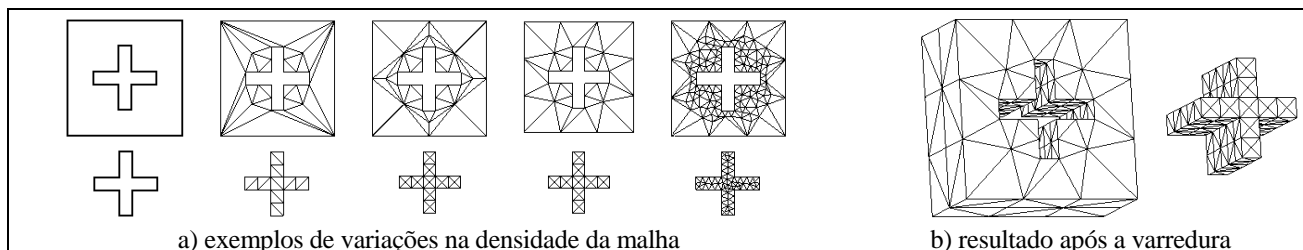



Figura VII.9 – Variações na densidade de malha devido à manipulação do número de pontos sobre o perfil gerador

VII.3.3 – A Inclusão de Buracos

Com o objetivo de evitar a realização de operações computacionalmente caras, como as booleanas, e aumentar o poder descritivo do modelador, o perfil gerador empregado nas operações de varredura pode ser composto por mais de uma silhueta, o que permite construir sólidos contendo buracos, como descrito no item VII.2.3 e ilustrado na figura anterior. Um perfil composto pode ser criado diretamente no modelador, por meio da combinação de silhuetas previamente definidas. Como já explicado no item IV.3.7, independentemente de ser aberta ou fechada, a silhueta externa deve ser única e envolver todas as demais. De forma a garantir a geração de modelos válidos, as silhuetas internas são sempre fechadas, não podendo envolver ou interceptar outras silhuetas do perfil. Opcionalmente, um perfil composto poderá ser obtido de um grafo PSLG, lido de um arquivo ".POL".

Para uma silhueta interna de um perfil composto, a malha resultante da varredura define um buraco, que é gerado da mesma forma que a malha da silhueta externa, porém com inversão na seqüência de pontos da silhueta, o que garante a orientação contrária das faces laterais do buraco. Em modelos gerados por varredura rotacional parcial ou translacional, além das superfícies laterais resultantes da varredura, fazem parte do modelo as faces planares definidas pelo perfil gerador, sobre as quais também é gerada, com o auxílio do programa *Triangle*, uma malha superficial compatível com a malha lateral obtida pela varredura. As figuras VII.10 e VII.11 mostram exemplos de modelos obtidos por varredura translacional e rotacional empregando perfis compostos.

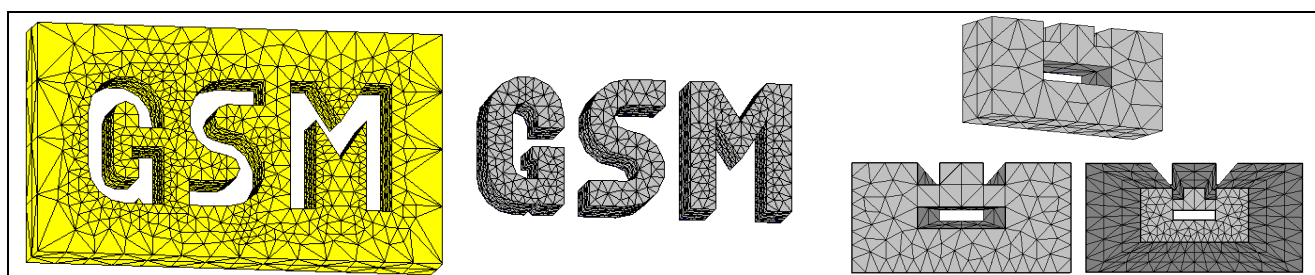



Figura VII.10 – Modelos obtidos com a varredura translacional de perfis compostos

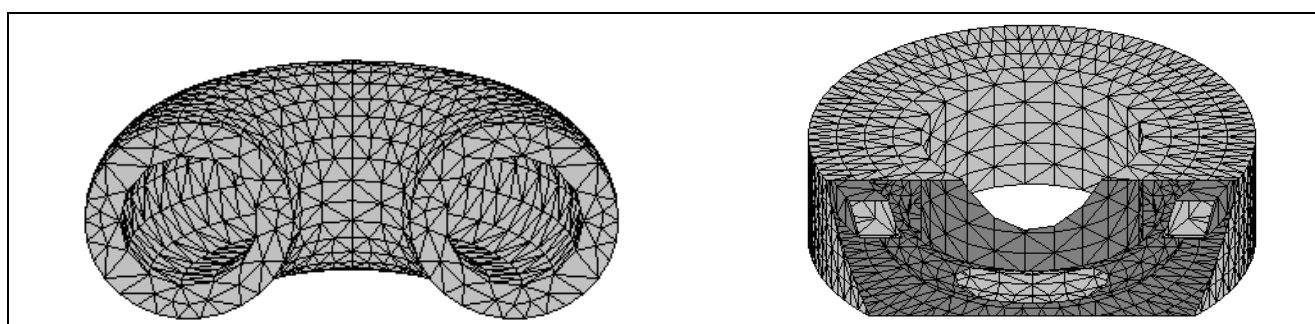



Figura VII.11 – Modelos obtidos com a varredura rotacional de perfis compostos

VII.4 – EXEMPLOS APLICADOS AO ELETROMAGNETISMO

Os exemplos apresentados nesta seção modelam dispositivos utilizados em projetos eletromecânicos e de instalações elétricas de baixa, média e alta tensão, como cabos, máquinas eletromagnéticas rotativas, transformadores e isoladores. O principal objetivo é mostrar a aplicabilidade do modelador a projetos que necessitam não só modelar componentes, mas também gerar, sobre as fronteiras definidas, uma malha de elementos finitos compatível e de boa qualidade, que possibilitará a simulação e análise correta dos componentes e seu interfaceamento, fundamental em aplicações eletromagnéticas.

VII.4.1 – O MODELAMENTO DE CABOS

O modelamento de cabos pode ser realizado por meio da combinação de círculos concêntricos sobrepostos, varridos sobre uma mesma trajetória linear, como ilustrado na figura VII.12. Representando um material condutor, a região central foi modelada a partir de um único círculo. Envolvendo a região central, as camadas de isolamento e proteção são modeladas a partir do anel resultante da diferença entre dois círculos concêntricos contendo raios distintos.

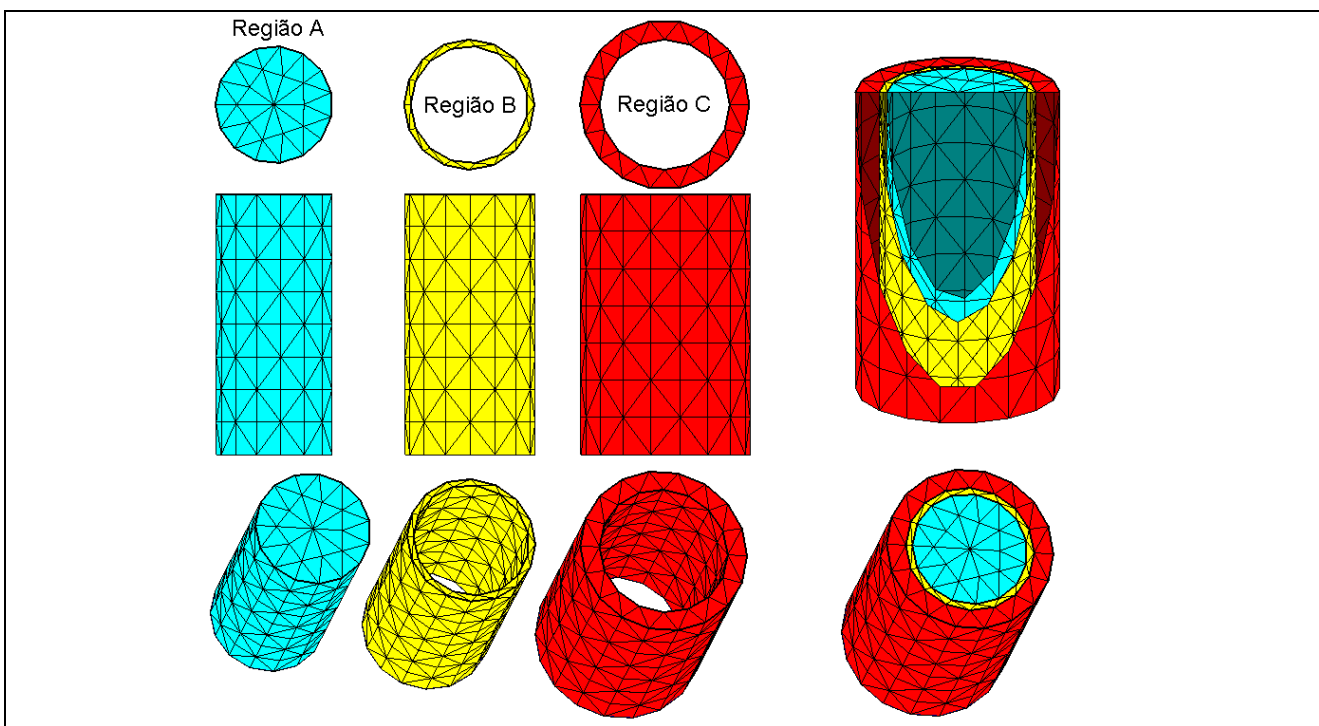



Figura VII.12 – O modelamento de um cabo simples

Visando à compatibilidade da malha superficial gerada sobre cada componente, o modelador permite ao usuário controlar a densidade da malha, por meio da especificação de parâmetros como a tolerância a ser aplicada na aproximação, o número de divisões atribuída ao segmento padrão e o número de níveis a ser gerado pela varredura. É possível notar, na figura anterior, que existe uma correspondência entre a forma, a posição e o tamanho dos elementos gerados na fronteira de cada componente a ser sobreposto. Tal correspondência é fundamental pois, além de assegurar a compatibilidade da malha, permite acoplar os elementos e transformá-los em uso-faces de uma mesma face na estrutura B-rep, o que garante a interpretação de maneira única da interface entre os componentes.

A figura VII.13 apresenta o modelamento de um cabo de três condutores. Diversamente do exemplo anterior, a região central é agora formada por um material isolante envolvendo os três condutores. O perfil utilizado na definição dessa região é formado por um círculo maior, do qual são retirados três círculos menores, correspondentes às seções transversais de cada condutor. Para garantir a compatibilidade, a malha superficial gerada sobre cada condutor deverá corresponder à malha gerada sobre seu buraco, na região central.

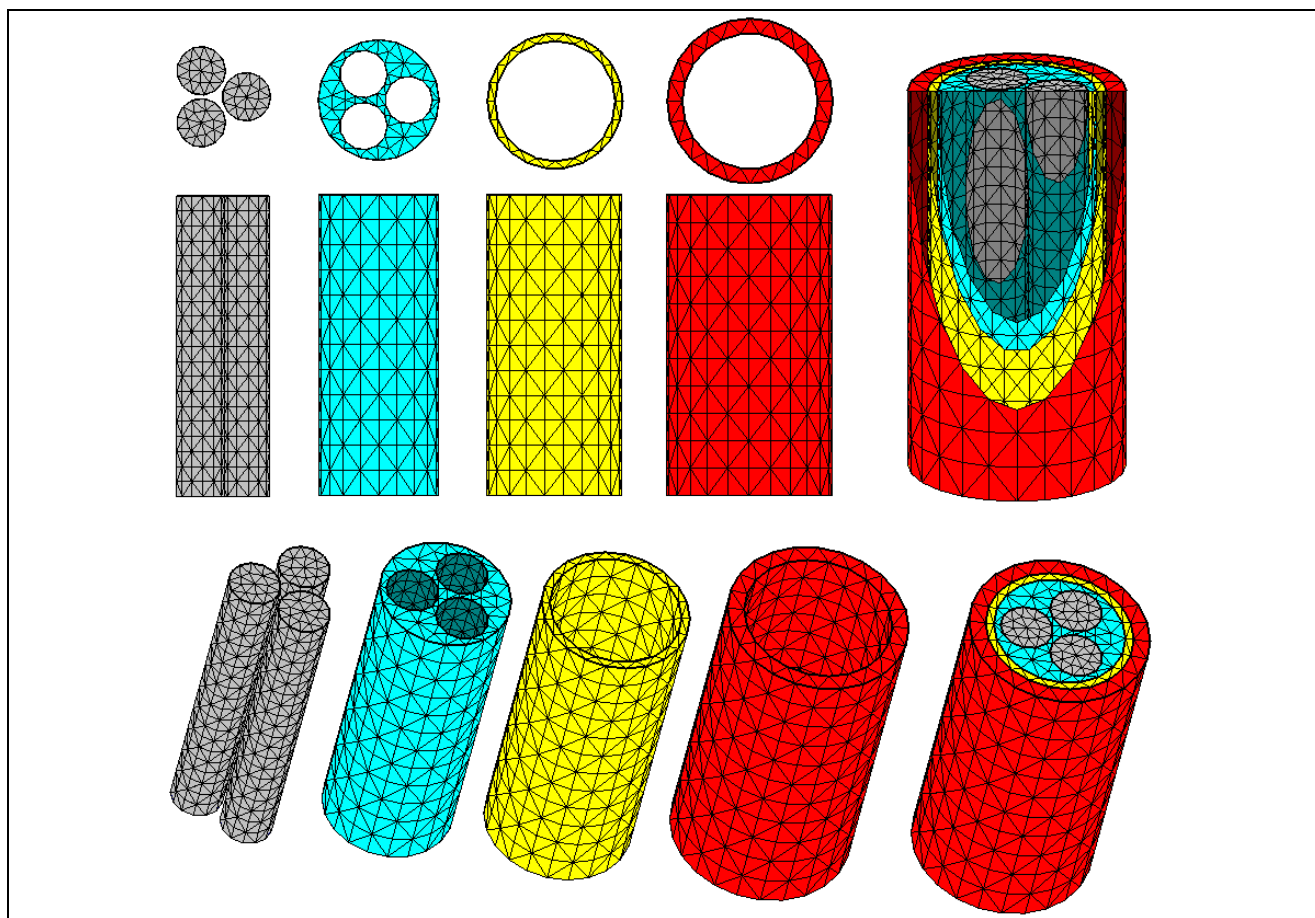



Figura VII.13 – O modelamento de um cabo composto por 3 fios

VII.4.2 – O MODELAMENTO DE MÁQUINAS ELÉTRICAS

Os exemplos apresentados a seguir modelam dispositivos utilizados em sistemas de conversão eletromecânica de energia, freqüentemente encontrados em motores e geradores. O primeiro exemplo modela um motor de corrente contínua simplificado. A figura VII.14 apresenta o processo de construção do modelo seguindo a abordagem CSG, o que requer um grande esforço computacional. O emprego de perfis geradores compostos, como explicado no item VII.3.3, facilita a construção da malha superficial, o que pode ser verificado nas figuras VII.15 e VII.16.

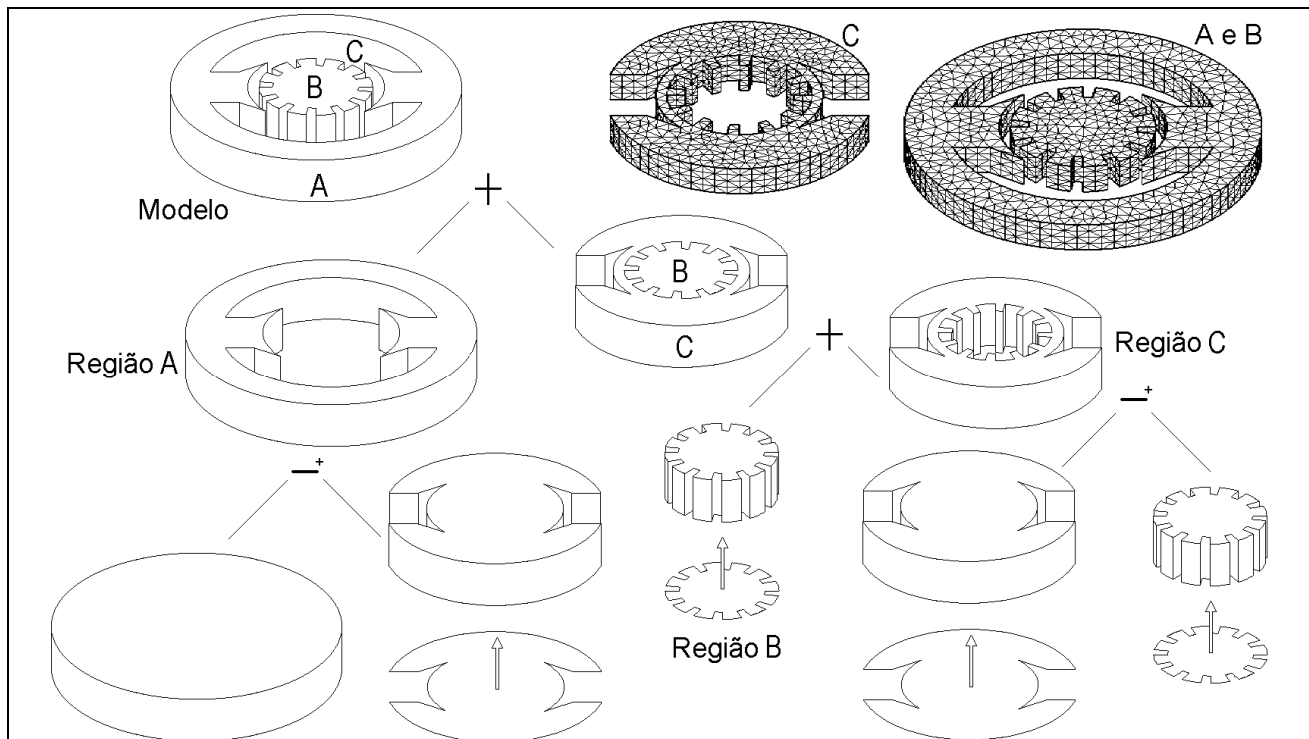



Figura VII.14 – Árvore CSG para a construção de um motor de corrente contínua simplificado

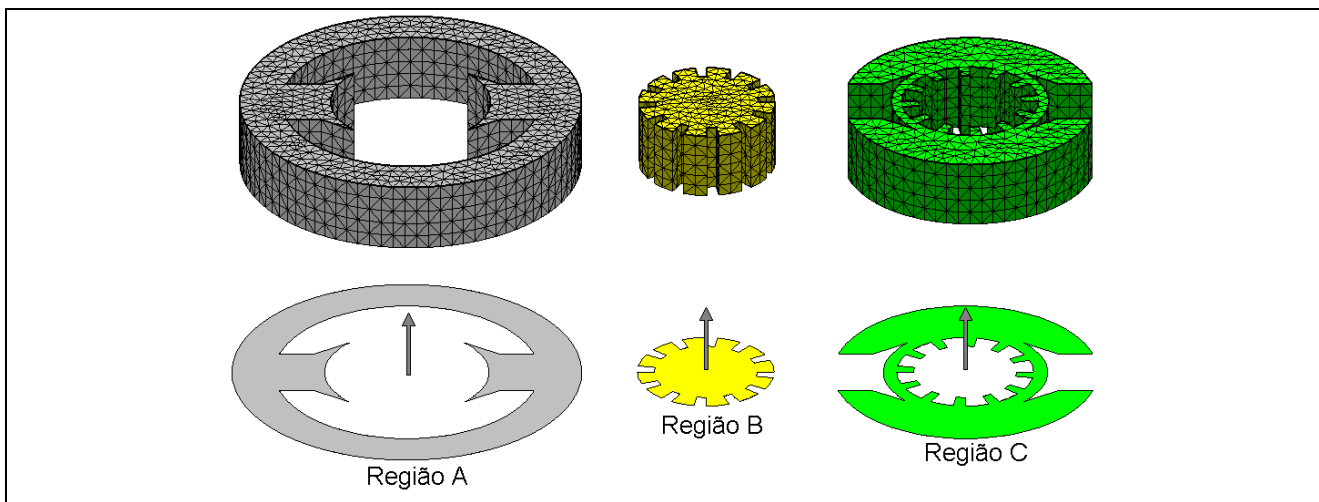



Figura VII.15 – Construção do motor anterior a partir de perfis compostos

a) Malha superficial gerada sobre o conjunto rotor-estator b) compatibilidade da malha entre as regiões

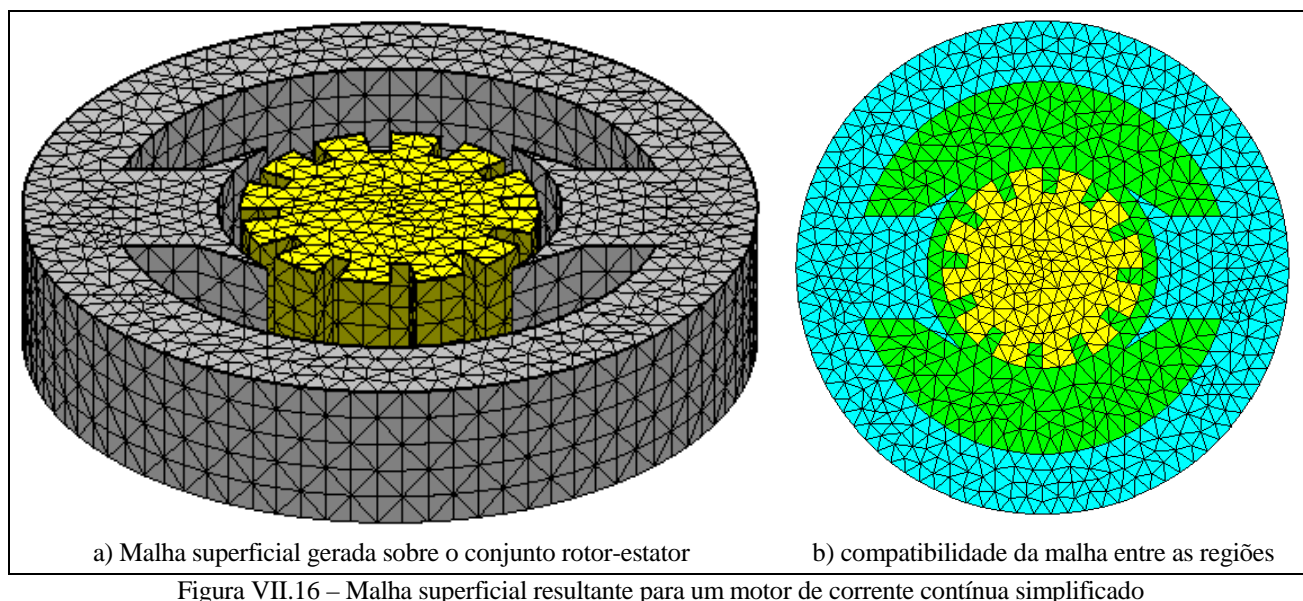



Figura VII.16 – Malha superficial resultante para um motor de corrente contínua simplificado

Outros dois modelos de máquinas elétricas um pouco mais complexos, propostos em [Del75], foram construídos utilizando o modelador. Ambos modelam máquinas de corrente contínua de dois pólos. As figuras VII.18 e VII.19 detalham o processo de construção empregado na obtenção de cada região, acompanhado das respectivas malhas superficiais. É possível notar, nessas figuras, a compatibilidade existente entre as malhas superficiais geradas pelo processo de varredura. A figura VII.17 destaca a compatibilidade entre as malhas obtidas sobre os perfis geradores.

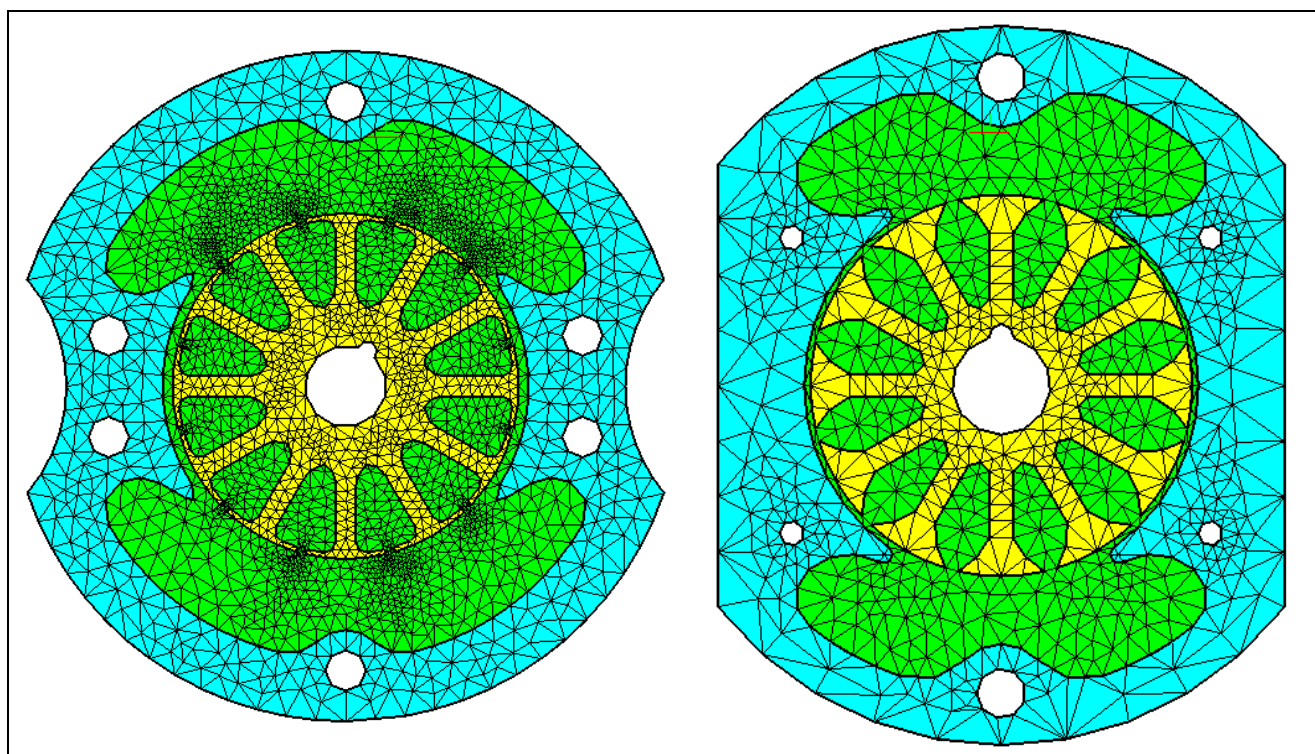



Figura VII.17 – Compatibilidade existente entre as malhas obtidas sobre os perfis geradores

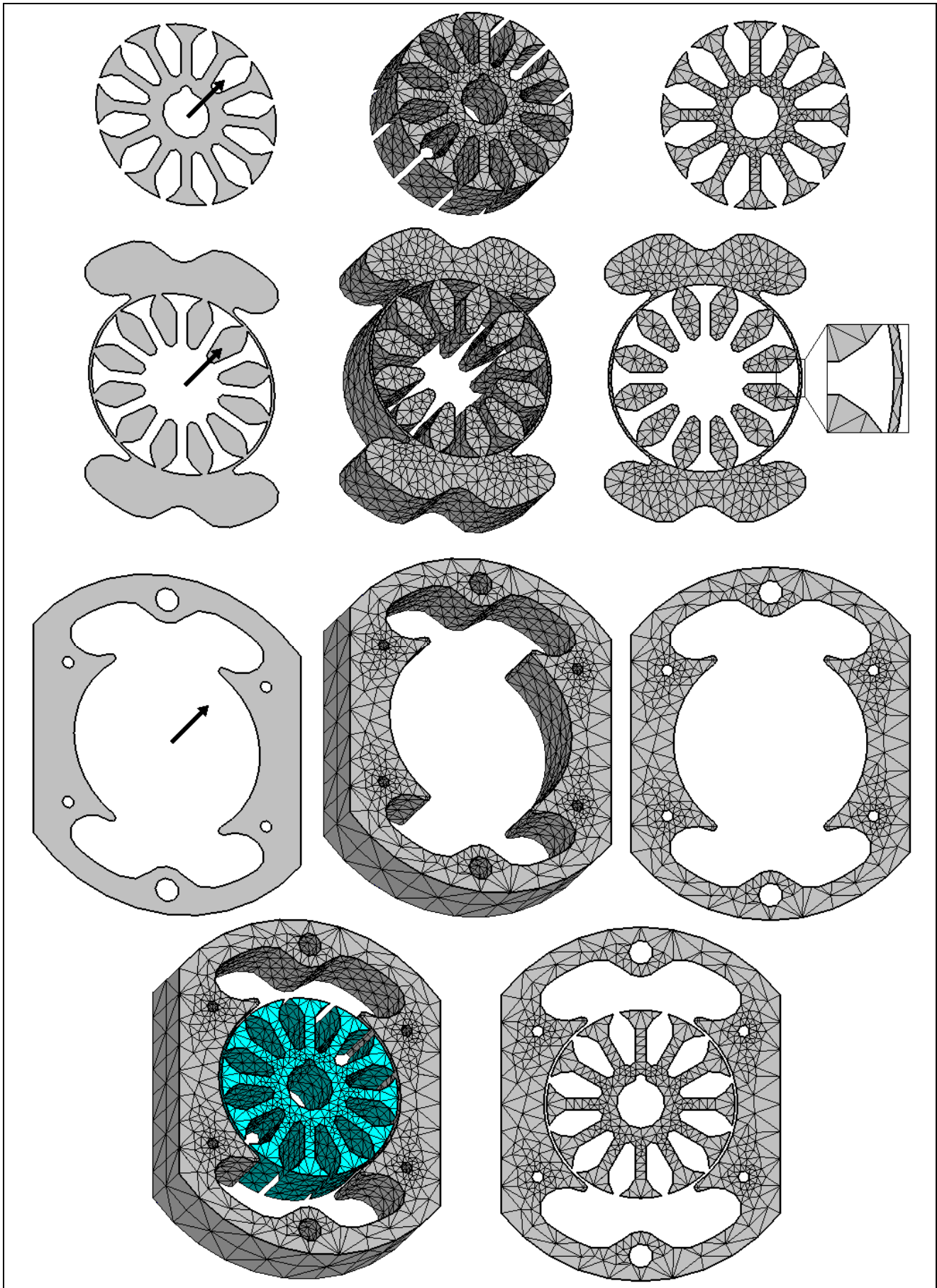



Figura VII.18 – Malha superficial obtida para o primeiro exemplo de máquina de corrente contínua de dois pólos

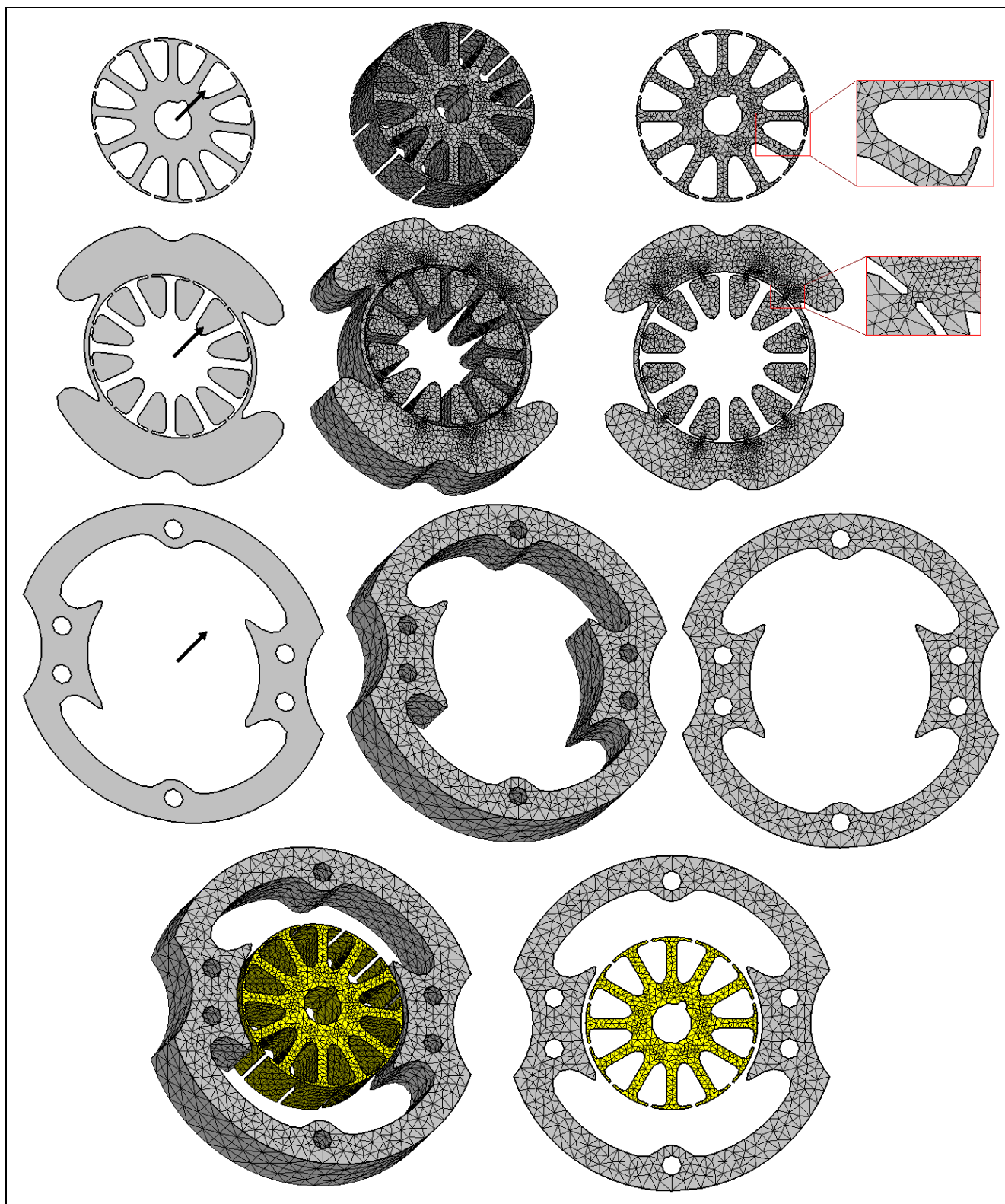



Figura VII.19 – Malha superficial obtida para o segundo exemplo de máquina de corrente contínua de dois pólos

VII.4.3 – O MODELAMENTO DE ISOLADORES

Utilizados em projetos de transmissão e distribuição de energia elétrica, os isoladores são equipamentos essenciais para o funcionamento de linhas de transmissão. De uma forma geral, sua função consiste em isolar o cabo energizado da parte aterrada do sistema. Possuem formato geralmente axi-simétrico, o que permite defini-los por varredura rotacional. A maioria é gerada a partir de perfis abertos, rotacionados 360°, como o isolador cerâmico de alta tensão [Lea79] ilustrado na figura VII.20. A varredura parcial apresentada na figura possui caráter apenas ilustrativo, não sendo normalmente utilizada em projeto de isoladores.

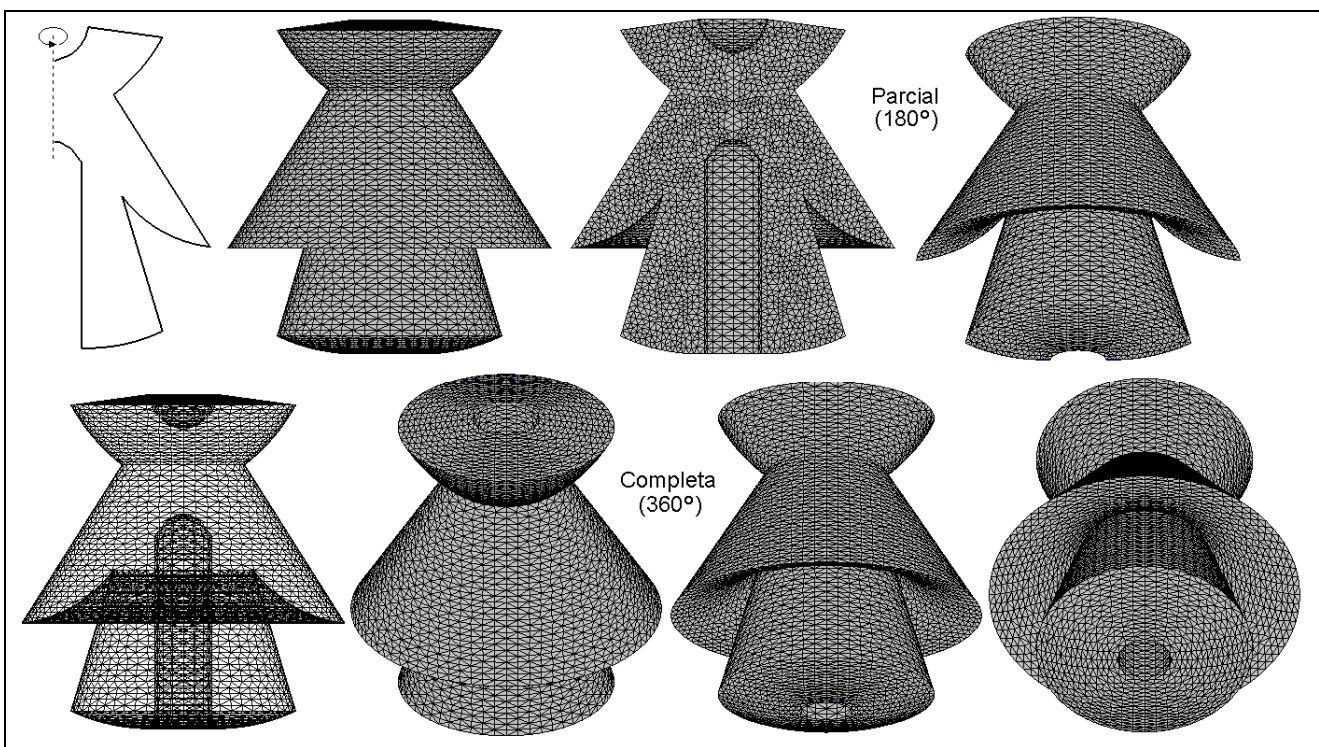



Figura VII.20 – Exemplo de malha superficial obtida para um isolador cerâmico de alta tensão

A figura VII.21 apresenta um outro exemplo de isolador em forma de pino, empregado em linhas de transmissão de 6 a 10 KV [Raz82]. Realiza-se também aqui a varredura rotacional de um perfil aberto. Obtém-se um modelo com formas aparentemente bem arredondadas, apesar de aproximado por facetas planares. Já a figura VII.22 modela um isolador especial para atmosfera contaminada [Raz82], composto por diversas camadas sobrepostas. A superior é definida pela varredura rotacional de um perfil fechado, enquanto as demais varrem um perfil aberto. O modelador permite controlar o número de setores gerados na varredura rotacional, de forma a compatibilizar a malha gerada entre dois perfis.

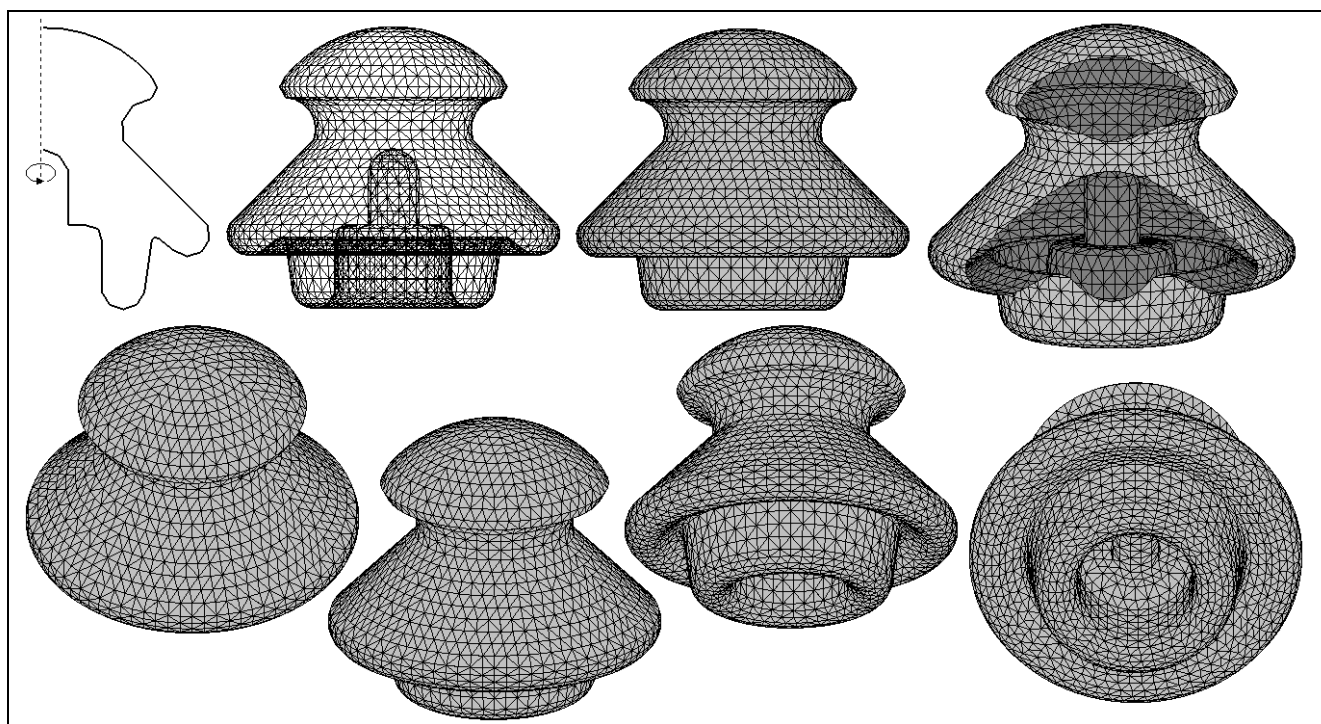



Figura VII.21 – Exemplo de malha superficial obtida para um isolador em forma de pino

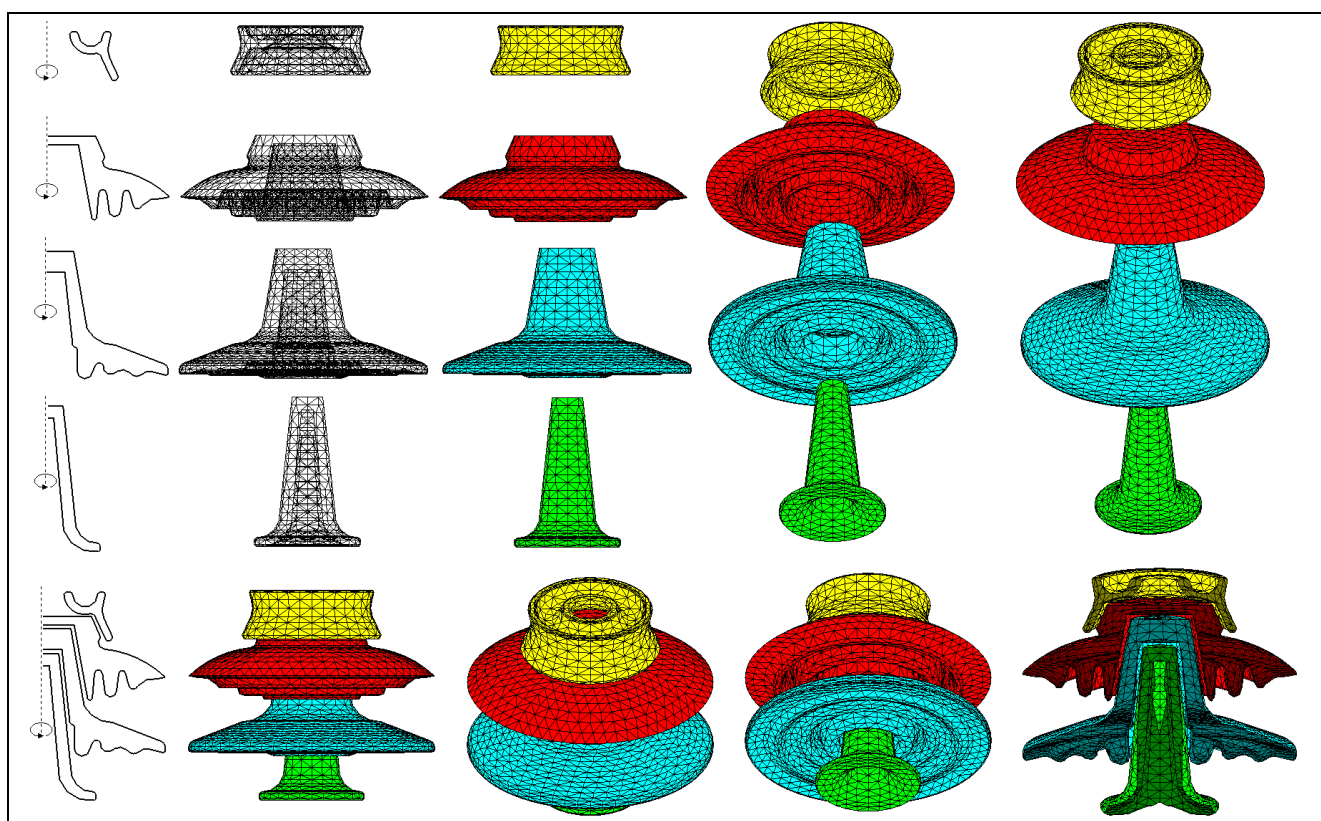



Figura VII.22 – Exemplo de malha superficial obtida para um isolador específico para atmosfera contaminada

VII.4.4 – O Modelamento de Bobinas

O modelamento de bobinas envolvendo núcleos magnéticos pode ser facilmente realizado pelo modelador. Um exemplo básico é apresentado na figura VII.23. O núcleo é definido pela varredura translacional de um quadrilátero – um quadrado, neste exemplo. Possuindo espessura constante, a bobina é definida pela diferença entre dois quadriláteros com bordas arredondadas, que possuem o mesmo formato mas dimensões diferentes.

A figura VII.24 apresenta um outro tipo de bobina, agora circular, contendo um núcleo cuja seção possui o formato de uma cruz com 4 dentes [Sep67], o que aumenta a área do núcleo e reduz a perda por dispersão. A bobina poderia ser gerada por varredura rotacional, porém para assegurar a compatibilidade entre a malha da bobina e a do núcleo, optou-se por gerá-la de forma translacional, incluindo, no perfil gerador, os pontos de contato com o núcleo.

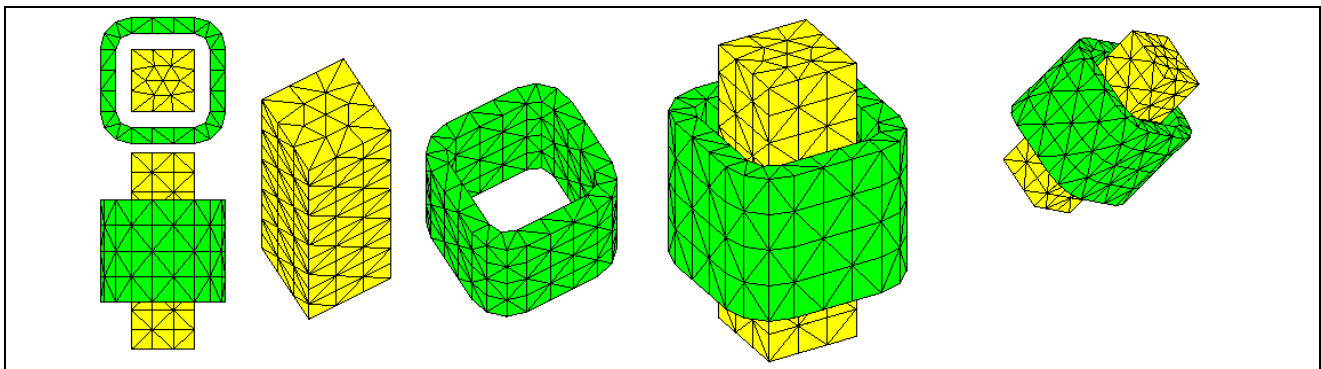



Figura VII.23 – Exemplo de malha superficial obtida para um conjunto bobina-núcleo

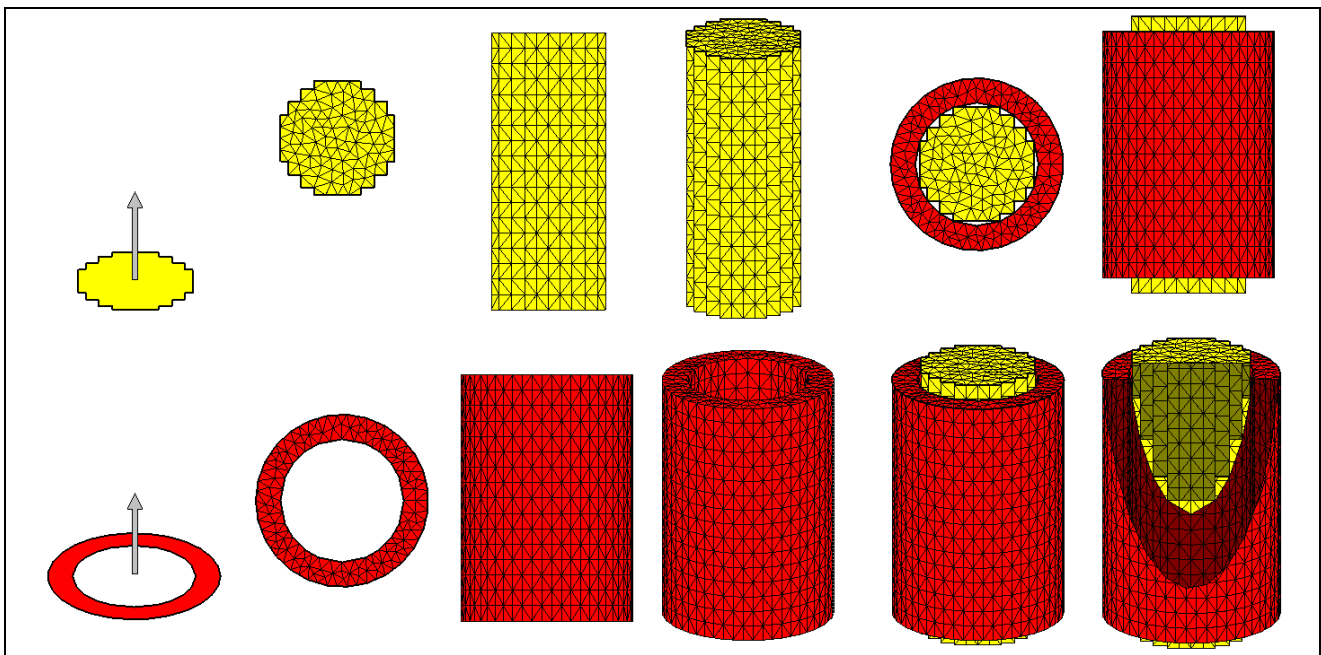



Figura VII.24 – Exemplo de malha superficial obtida para um conjunto bobina – núcleo em forma de cruz

VII.4.5 – O MODELAMENTO DE TRANSFORMADORES

Embora não sejam propriamente dispositivos de conversão eletromecânica de energia, os transformadores constituem um dispositivo auxiliar importante no problema global de conversão de energia, estando sua análise, em muitos aspectos, intimamente relacionada à dos motores e geradores.

O modelamento de transformadores é um pouco mais complexo que os exemplos anteriormente apresentados. A figura VII.25 apresenta um exemplo do processo de construção de um transformador seguindo a abordagem CSG. Uma vez que as operações booleanas não estão ainda disponíveis no modelador, será necessário defini-lo utilizando perfis compostos e operações de varredura, tomando todo cuidado para que a malha resultante de seus componentes seja compatível.

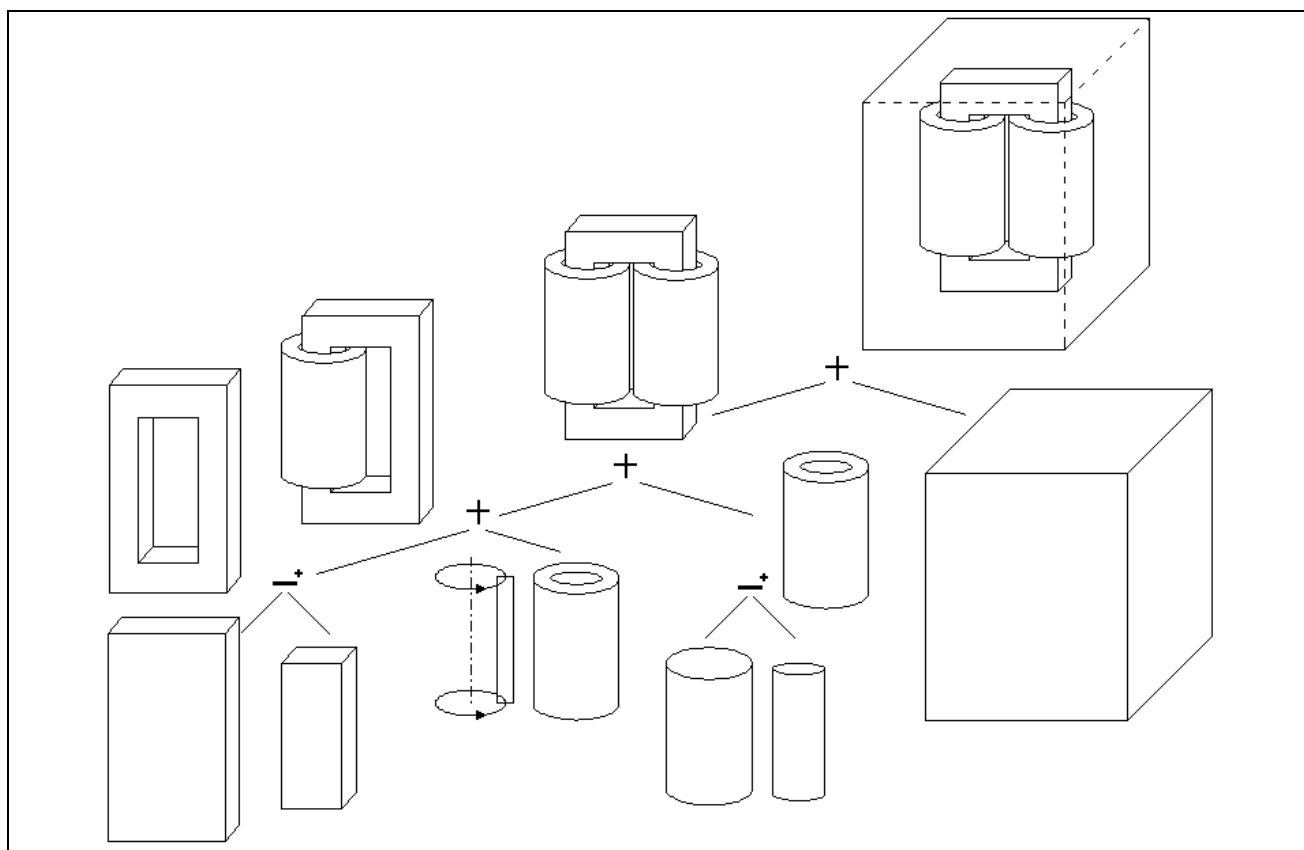



Figura VII.25 – Exemplo de árvore CSG para a construção de um transformador bifásico

A primeira idéia que surge é a de definir o núcleo utilizando um perfil composto, semelhante ao apresentado na figura VII.26. Esta estratégia, porém, levaria à obtenção de uma malha sobre o perfil – gerada pelo *Triangle* – que não seria compatível com a obtida no interior da bobina – gerada por varredura translacional. Não dispondo ainda das operações booleanas para realizar o acerto entre as malhas, a estratégia escolhida foi a de gerar o núcleo pela composição de várias partes – todas elas obtidas por varredura translacional – fazendo coincidir a malha lateral obtida em cada parte.

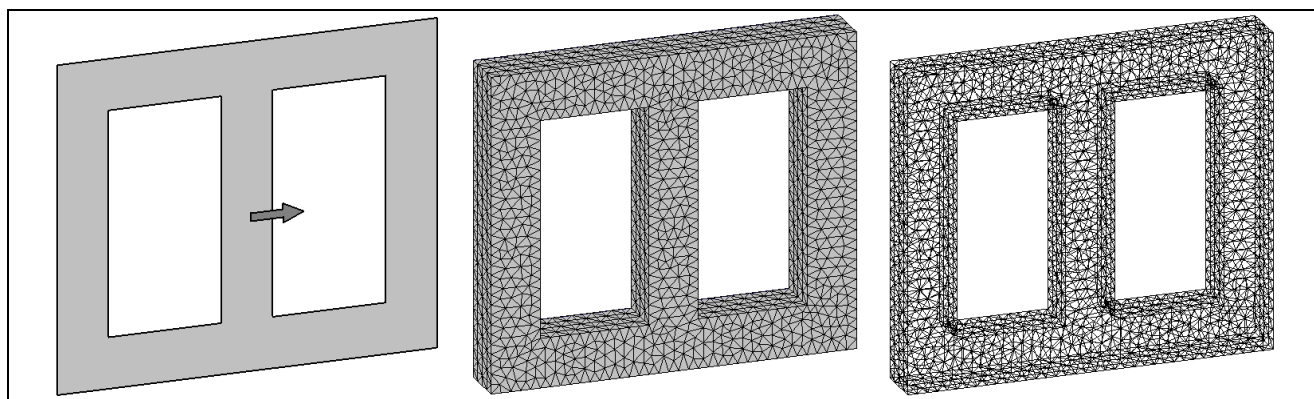



Figura VII.26 – Geração do núcleo por varredura translacional de um perfil composto

A figura VII.27 elucida o processo empregado. Cada uma das bobinas é obtida pela varredura translacional do perfil definido. As faces laterais são geradas e triangularizadas pela varredura, enquanto o *Triangle* discretiza a face superior e a inferior da bobina. As faces correspondentes ao buraco são geradas pela varredura. O núcleo é definido pelos componentes apresentados, também obtidos por varredura. A malha gerada entre esses componentes é compatível, sendo posteriormente removida por meio dos operadores de Euler para colagem de regiões e faces (*Glue Regions* e *Glue Faces*), já disponíveis no modelador, como mostrado na figura VII.28.

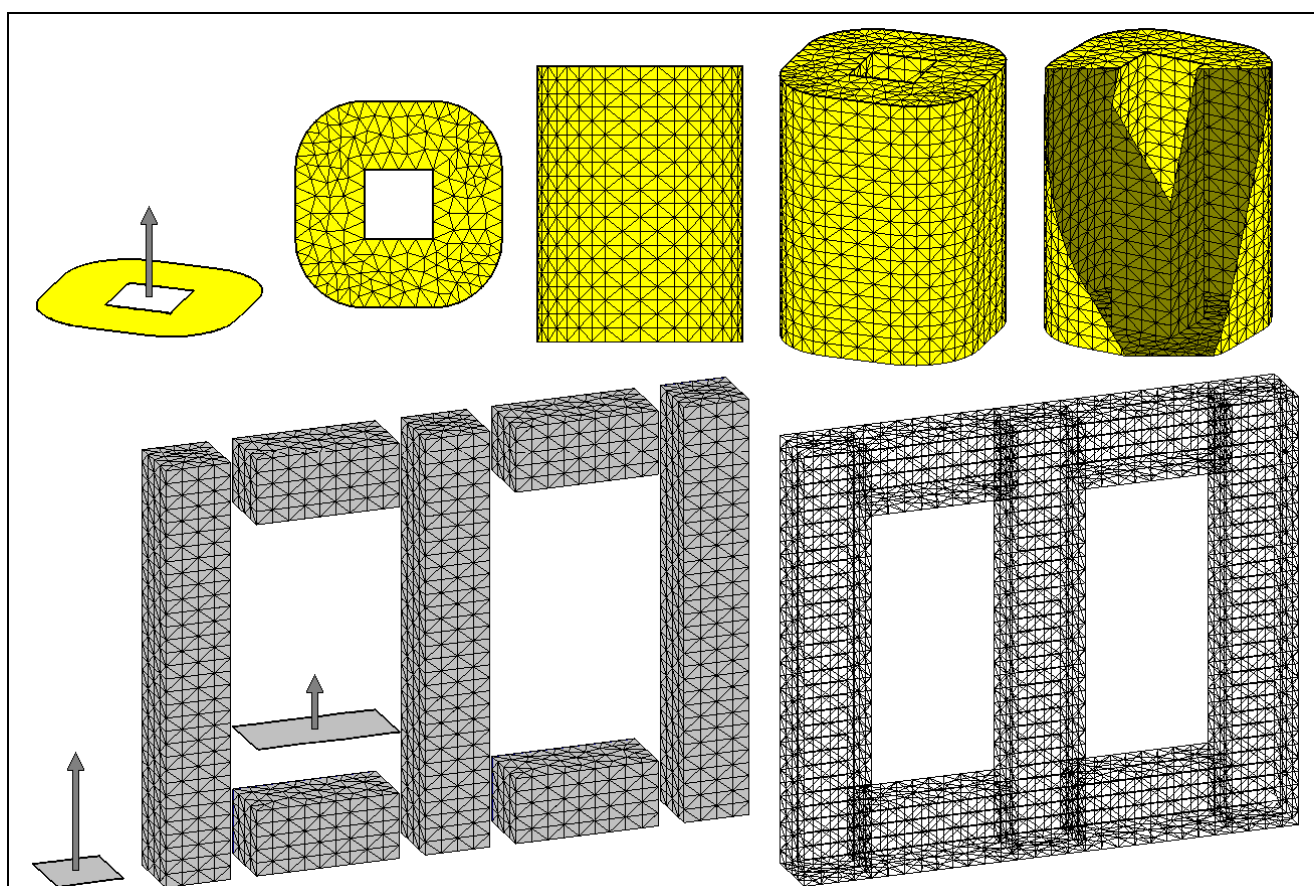



Figura VII.27 – Processo de geração de malha empregado na construção do transformador

As faces que formam o buraco da bobina podem ser montadas com suas correspondentes no núcleo e transformadas em uso-faces de uma mesma face, por meio do operadores de montagem (*Assemble Faces* e *Assemble Regions*). O modelo resultante é apresentado na figura VII.29. Opcionalmente, o modelo gerado poderá ser envolvido por uma caixa representando a carcaça do transformador, possibilitando também a simulação e análise de sua interface com o meio dielétrico.

Para a construção desse modelo foi necessário utilizar diversos planos de trabalho, uma vez que os perfis geradores das bobinas e dos componentes que definem o núcleo encontram-se posicionados em planos diferentes no espaço. A redefinição do plano de trabalho é uma tarefa simples, poderosa e prática, disponível no modelador.

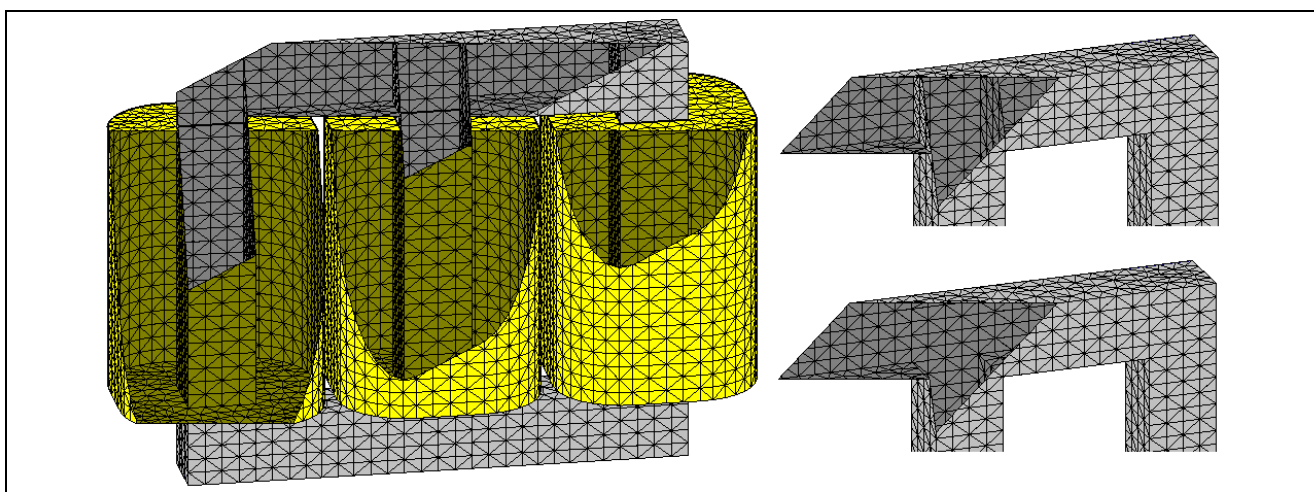



Figura VII.28 – Detalhamento da colagem e montagem de faces

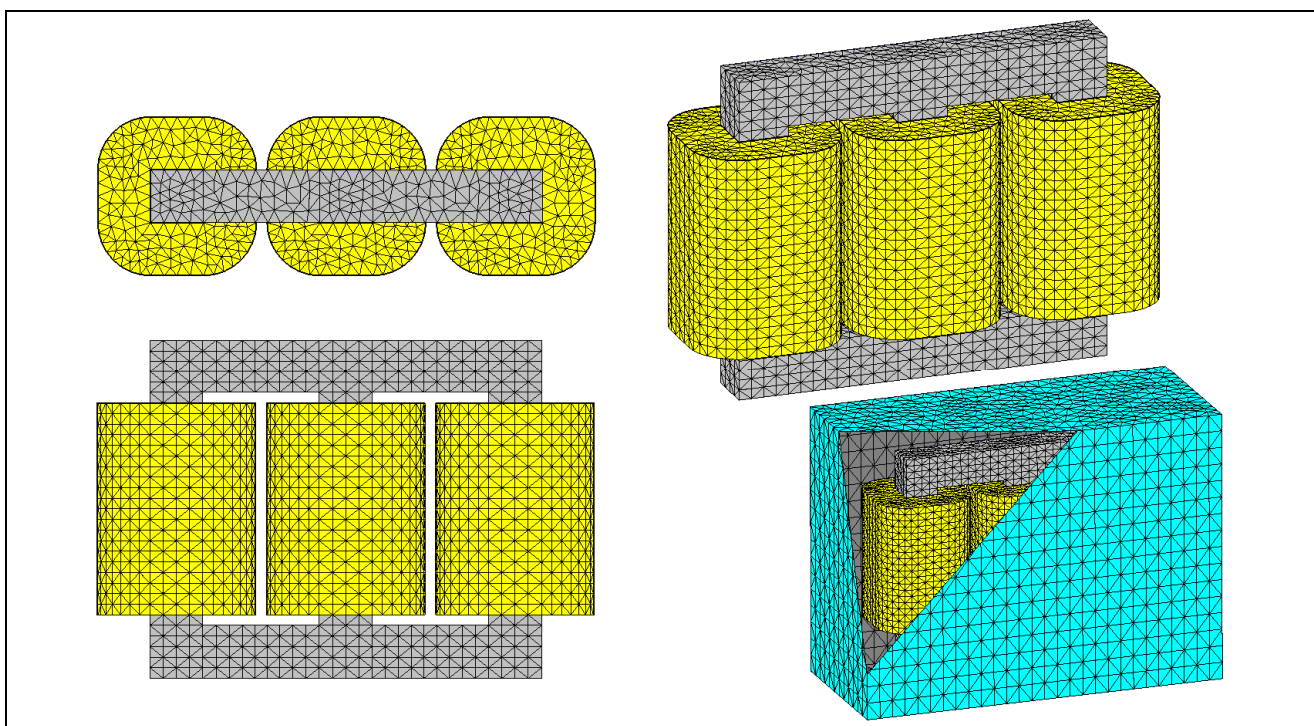



Figura VII.29 – Visão geral do transformador

VII.4.6 – OUTROS MODELOS DE INTERESSE

O exemplo a seguir detalha um divisor de tensão, utilizado em sistemas de medição para amostragem de valores de tensão em determinados pontos de um circuito elétrico. Ele foi gerado por varredura rotacional de perfis abertos e fechados, como mostra a figura VII.30. É importante notar a compatibilidade existente entre a malha superficial obtida para cada componente, controlada pela especificação do número de setores resultantes da varredura rotacional.

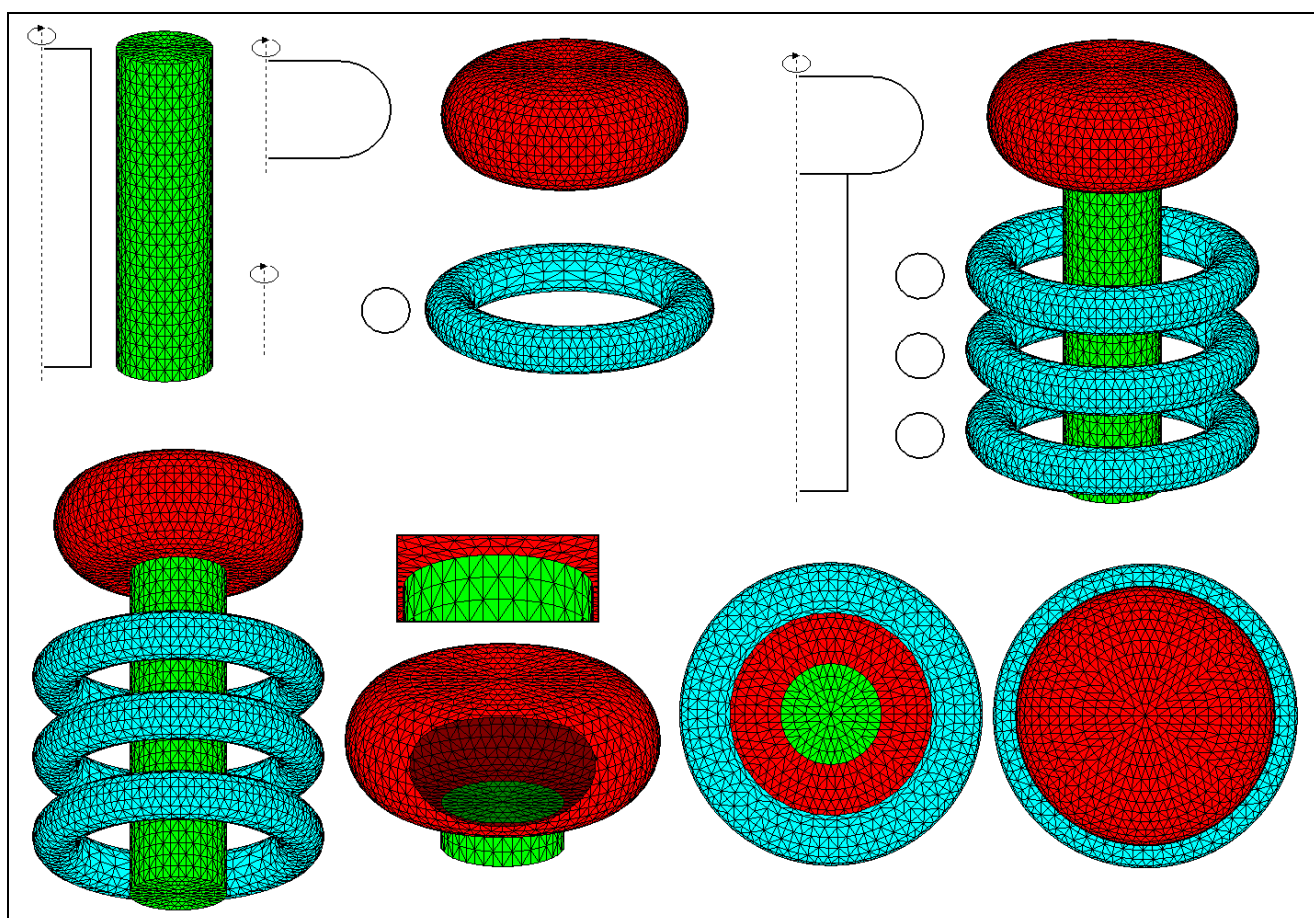



Figura VII.30 – Malha superficial obtida para um divisor de tensão

"Por vezes é penoso cumprir o dever;
mas nunca é tão penoso como não cumpri-lo."
Alexandre Dumas

VIII – CONCLUSÃO

Neste capítulo é feita uma síntese final do trabalho, apresentando reflexões acerca do seu teor, dos resultados obtidos e de seu alcance. Após algumas ponderações em relação ao conteúdo, é avaliada a situação atual do projeto de desenvolvimento do modelador, discutindo-se alguns de seus aspectos mais relevantes. A seguir, são destacadas as suas principais contribuições e sugeridos aperfeiçoamentos que poderão ser desenvolvidos em futuros trabalhos correlatos.

VIII.1 – REFLEXÕES ACERCA DESTE TRABALHO, SEU CONTEXTO E RESULTADOS

A evolução da modelagem de sólidos está intimamente relacionada ao desenvolvimento de aplicações CAD/CAM e caminha em busca de modelos dotados de todas as informações necessárias para a automação de processos, do projeto à manufatura. Nessa direção, uma das principais aplicações de modeladores de sólidos tem sido a análise em engenharia, que verifica a consistência do modelo e a sua compatibilidade com o ambiente onde será realmente utilizado, podendo justificar a aprovação ou a necessidade de alteração de um projeto. O processo de análise percorre as seguintes etapas: pré-processamento, que define a geometria e as características físicas do modelo e gera a malha de elementos finitos; processamento, que monta e resolve os sistemas de equações empregados; pós-processamento, que extrai e sintetiza as informações significativas sobre o modelo. Conforme explicado no capítulo I, esta tese de doutorado consistiu em proceder ao desenvolvimento de um sistema de modelagem de sólidos a ser utilizado como pré-processador na resolução de problemas eletromagnéticos pelo Método de Elementos Finitos.

Constituindo uma das principais ferramentas utilizadas na simulação de problemas de engenharia – e em especial, de problemas eletromagnéticos –, o Método de Elementos Finitos encontra-se hoje bem sedimentado e formalizado, tendo atingido um grau de maturidade suficiente para justificar o desenvolvimento de um bom número de softwares de apoio ao projetista. Ele deve ainda evoluir no sentido de adaptar-se a cada tipo de aplicação, permitindo uma interação cada vez maior com o usuário. Num futuro próximo, espera-se que o Método de Elementos Finitos torne-se transparente para o usuário, que poderá, durante a concepção e projeto do produto, obter instantaneamente uma resposta visual ilustrando seu comportamento diante de todas as modificações geométricas ou físicas introduzidas.

O emprego de poderosas técnicas interativas gráficas durante a concepção e projeto do produto – como as desenvolvidas para o Subsistema de Interface e apresentadas no capítulo III – constitui um dos mais modernos aliados ao Método de Elementos Finitos. Deve-se, porém, conciliar a concepção de cada produto com as especificidades de seu domínio de aplicação. Para isso, a área de eletromagnetismo carecia de um modelador capaz de definir componentes sólidos manufaturáveis que pudessem ser acoplados entre si gerando fronteiras internas e externas representadas de maneira única, de forma a garantir a compatibilidade da malha de elementos finitos gerada. O trabalho aqui proposto visa suprir exatamente essa carência, mediante o desenvolvimento de um modelador de sólidos que atenda a este e a outros requisitos relacionados no capítulo II, e que possa ser posteriormente acoplado a um gerador auto-adaptativo de malhas volumétricas.

Além do uso de técnicas interativas gráficas eficientes, outro requisito importante relacionado à Interface é a facilidade de utilização do modelador pelo usuário. Os softwares de elementos finitos têm tudo a ganhar com a melhoria das interfaces e dos recursos de modelagem. Problemas referentes à portabilidade estão sendo resolvidos graças à adoção de bibliotecas gráficas portáveis – como a OpenGL – e de padrões para o intercâmbio de dados gráficos – como o uso de arquivos neutros e da padronização STEP.

Do ponto de vista do usuário, a melhor abordagem para manipulação de um modelo é a construtiva. Integrante do Subsistema de Modelagem e descrito no capítulo IV, o CSG é um esquema de representação bastante difundido, devido principalmente à existência de um embasamento matemático rigoroso e de fácil compreensão, ao conhecimento dos algoritmos envolvidos e à ampla área de aplicação. Possui poderosas técnicas de edição, além de manter o registro do processo de construção, o que facilita e agiliza a realização de modificações. Os objetos representados são manufaturáveis, uma vez que são formados por operações bem definidas sobre sólidos reais. Estes fatores fazem com que a representação CSG possua importância considerável na manufatura industrial.

Entretanto, do ponto de vista da aplicação, a abordagem por fronteira é mais eficiente. Além do CSG ser um modelo não avaliado, necessitando gerar o sólido todas as vezes em que for utilizado, a abordagem por fronteira dispõe dos elementos de representação que o modelador deve manipular e apresentar, essenciais também para a geração da malha de elementos finitos, possuindo ainda métodos poderosos, não implementáveis na abordagem construtiva. Daí o interesse pela construção de modeladores híbridos, capazes de manter duas ou mais representações equivalentes. Se por um lado o problema de computar a representação B-rep a partir da CSG está bem compreendido, por outro, o problema inverso de conversão da representação B-rep para CSG não

está resolvido de maneira genérica e levanta questões ainda pouco exploradas, que possuem interesse teórico significativo. Sendo assim, os modeladores de sólidos atuais que mantêm essas duas representações são, na realidade, multi-representacionais, e não híbridos.

As principais características da representação por fronteira foram abordadas no capítulo V, que além de descrever a implementação do Subsistema de Representação, mostrou as estruturas de dados B-rep mais importantes e detalhou as operações de construção de modelos fundamentais nessa representação. Vale a pena ressaltar que a estrutura de dados B-rep definida para este trabalho permite a **composição de sólidos de variedade simples**, garantindo a geração de sólidos manufaturáveis, característica aparentemente não contemplada pelas estruturas de dados descritas na literatura. Tal estrutura combina aspectos importantes presentes na proposta de Mäntylä para sólidos de variedade bidimensional [Män88] e na proposta de Muuss e Butler para sólidos de variedade múltipla [Muu91]. Da proposta de Mäntylä, mantém-se a orientação fornecida pelas semi-arestas e a possibilidade de existência de anéis e cavidades. Por outro lado, descarta-se a inconveniente necessidade de uma aresta ser compartilhada por exatamente duas faces. Tal necessidade, porém, continua a ser válida em um novo contexto: cada uso de uma aresta deve ser compartilhado por exatamente dois usos de face, dentro de um mesmo componente do modelo, que continua a ser de variedade bidimensional. Da proposta de Muuss, aproveita-se a estrutura hierárquica, a idéia de separar a definição das entidades de suas ocorrências – face e uso da face, arestas e uso da aresta, etc. – e a possibilidade de várias faces compartilharem uma mesma aresta, o que é fundamental na definição de fronteiras internas. Da mesma forma, descarta-se a restrição quanto a todos os usos da face pertencerem a uma única casca e a possibilidade de incluir componentes pendentes e de menor dimensão, que não seriam manufaturáveis.

Para que um objeto descrito pela representação CSG possa ser tratado com os propósitos de verificação e análise, é necessário que ocorra uma conversão para a representação de sua fronteira, também descrita no capítulo V. Essa conversão é realizada por um procedimento denominado avaliador de fronteira. A partir da malha gerada sobre as primitivas, esse procedimento identifica e define as interseções entre as primitivas e, a seguir, localiza e valida todos os componentes que refletem o formato final do objeto resultante. Obtém-se assim uma malha consistente, não redundante, que descreve toda a fronteira do modelo sólido.

Uma vez que a malha superficial do objeto CSG resultante é obtida a partir da malha superficial gerada sobre a primitiva, é importante garantir a qualidade da malha inicialmente gerada sobre a primitiva, o que é realizado pelo Subsistema de Geração de Malha, descrito no capítulo VI. Além disso, é importante que se tenha uma boa aproximação do sólido pela malha e que exista uma correspondência entre os elementos da fronteira do sólido com os elementos da fronteira da malha.

Em geral, a solução obtida usando o método de elementos finitos somente pode ser uma boa aproximação da solução analítica se a malha for uma boa aproximação do domínio do problema. É desejável, portanto, que a geometria da malha se aproxime da geometria do sólido da forma mais precisa possível, fornecendo um tamanho adequado para os elementos. Espera-se que em um futuro próximo, a função de discretização, que aparece ainda como uma das tarefas computacionalmente mais caras, seja atenuada em proveito de outros benefícios, como o incremento da complexidade dos problemas tratados devido à introdução de não-linearidades, acoplamentos, etc.

Além de elucidar conceitos e ilustrar a potencialidade do modelador com exemplos, o capítulo VII discutiu a necessidade e a geração da base de dados neutra, fundamental para o intercâmbio de informações com outros aplicativos relacionados, principalmente processadores e pós-processadores.

Este foi o primeiro trabalho desenvolvido no GOPAC enfocando a modelagem computacional de sólidos. Ele deverá ser integrado a outros trabalhos que, reunidos, formarão um produto mais completo para cálculo de campos eletromagnéticos. Além de um assunto relativamente novo, esta abordagem agrega conhecimentos de três áreas diferentes: a engenharia elétrica, a geometria computacional e a computação gráfica. As duas últimas não são tão familiares aos integrantes do GOPAC, motivo pelo qual o texto mereceu um maior nível de detalhamento, necessário para que se especificasse o que é entendido como um modelador de sólidos adequado para o eletromagnetismo e passível de construção pelo grupo. No decorrer do texto, o assunto limitou-se ao escopo deste trabalho, não perdendo de vista o objetivo final de um modelador mais completo. Espera-se que o material aqui apresentado seja útil para os integrantes do grupo – atuais e futuros –, fornecendo o conhecimento básico para o desenvolvimento de outros trabalhos relacionados à modelagem computacional de sólidos e CAD.

VIII.2 – Situação Atual do Modelador

Uma vez apresentadas as principais características de projeto e implementação desta versão do modelador, e visando ao seu desenvolvimento futuro, torna-se importante discutir alguns de seus aspectos do ponto de vista da implementação e utilização. Entre os aspectos de maior interesse, destacam-se os seguintes:

– integra o uso de vários esquemas de representação de sólidos, entre eles o instanciamento de primitivas, a varredura, as estruturas CSG e B-rep, além da representação fio-de-arame;

– armazena e recupera modelos em bases de dados permanentes (arquivos), permitindo ao usuário construir um modelo por etapas e não perdê-lo;

– cria e manipula perfis bidimensionais a partir de primitivas 2D, que podem ser combinadas entre si e sofrer transformações geométricas;

– permite gerar perfis compostos por mais de um contorno, o que acelera a construção de sólidos complexos e evita a aplicação de operações booleanas, que são computacionalmente caras;

– constrói primitivas bidimensionais também a partir de arquivos, o que possibilita definir, armazenar e modificar perfis complexos;

– instancia primitivas sólidas, bem como define outros sólidos por meio da varredura translacional ou rotacional de perfis abertos ou fechados, triangularizando a superfície resultante;

– verifica todos os parâmetros fornecidos pelo usuário antes que um comando seja executado;

– dispõe de facilidades para apontar e selecionar os componentes bi e tridimensionais do modelo, permitindo referenciá-los de forma direta;

– apresenta visões coloridas do modelo, bem como visões fio-de-arame, com ou sem remoção de linhas escondidas;

– possibilita ao usuário "caminhar" em torno e dentro do modelo, gerando sua imagem sob qualquer ponto de vista;

– possibilita definir, armazenar e recuperar propriedades físicas e dados relacionados ao cálculo, bem como atribuir tais informações aos componentes do modelo;

– disponibiliza arquivos que compõem a base de dados neutra, contendo informações sobre geometria, malha e materiais, o que propicia o intercâmbio de informações com outros aplicativos;

– a estrutura de dados está preparada para manipular polígonos com qualquer número de lados, contendo até mesmo buracos. Além disso, está em condições de incorporar faces e arestas curvas;

– a representação B-rep e a malha obtida estão intimamente relacionadas: na realidade, a fronteira gerada na representação B-rep representa a malha gerada sobre o modelo.

Outros aspectos refletem, na realidade, limitações do produto obtido. São eles:

– o sistema é ainda poliedral, ou seja, todas as curvas e superfícies são aproximadas por segmentos de reta e planos, respectivamente. A definição de uma curva por segmentos de reta eleva o número de elementos gerados na discretização, e o uso de uma resolução alta tende a deixar o sistema bastante lento;

– o modelador gera uma malha composta somente por elementos triangulares de primeira ordem. Elementos quadrangulares poderiam ter sido gerados, porém o uso de elementos triangulares torna o sistema mais eficiente, tanto do ponto de vista numérico quanto gráfico;

– permite a definição de sólidos ocupando uma mesma região do espaço. Se por um lado isso é desejável para a construção de modelos CSG, por outro não existe nenhum controle sobre o posicionamento final dos componentes no modelo;

– apesar da estrutura de dados permitir manipular polígonos com qualquer número de lados, a avaliação da fronteira desenvolvida até o momento restringe-se à manipulação de triângulos. Essa restrição impede o tratamento de outras formas poligonais, mas garante a obtenção de uma malha totalmente triangularizada como resultado;

– até o momento, as operações de varredura translacional somente podem ser realizadas seguindo caminhos lineares – utilizando um único vetor –, e as operações de varredura rotacional ocorrem somente em caminhos circulares;

– não podem ser realizadas, por enquanto, operações booleanas e de recorte sobre primitivas bidimensionais. Tais operações serão provavelmente incorporadas junto com as operações booleanas para objetos sólidos, que serão desenvolvidas em um trabalho de mestrado do GOPAC [Nun00];

– a não disponibilidade das operações boolenas para sólidos limita em muito o poder descritivo do modelador. Atualmente, fica a cargo do usuário garantir a compatibilidade da malha entre componentes, dificultando e restringindo a construção de modelos. Esta tarefa deverá ser realizada pelas operações booleanas, por meio da combinação e seleção dos elementos que compõem a malha de cada componente;

– não possui recursos para iluminação do modelo, limitando-se a apresentar faces coloridas, porém opacas. Não define fontes de luz nem utiliza técnicas de iluminação, sombreamento ou efeito de transparência.

Torna-se importante também analisar criteriosamente outro aspecto que pode ser observado no modelador atual: a junção entre a representação da fronteira e a representação da malha, que são representadas praticamente pela mesma estrutura – a B-rep. Essa junção traz como vantagem uma interpretação completa do modelo sob o ponto de vista da malha, mas por outro lado acaba perdendo a noção correta do que seria a face original e misturando o conceito de elemento e face. Mais inconveniente do que isso, essa junção acaba repetindo para todos os elementos da malha a noção de "uso" introduzida na representação B-rep. Assim, ao invés de somente as faces originais possuírem referência para os componentes face, uso-face, aresta, uso-aresta, vértice e uso-vértice do modelo B-rep, todos os elementos definidos em uma face passam a fazê-lo, tornando a estrutura ainda menos concisa e mais redundante. Isso diminui a eficiência do sistema, além de levar a um consumo maior do espaço para armazenamento – tanto em memória quanto em arquivo.

Essa solução de juntar a representação da fronteira com a representação da malha foi adotada apenas de forma provisória: como as superfícies originalmente curvas teriam que ser divididas em facetas poligonais planares – para a representação B-rep –, e em elementos triangulares – para a malha –, optou-se por gerar sempre facetas triangulares, que corresponderiam aos elementos da malha. Faces planares acabaram também seguindo essa estratégia e foram divididas, transformando toda a fronteira em elementos de malha, o que acabou dificultando a recuperação das faces originais e sobrecarregando a estrutura de dados definida, sem explorar seu potencial para definição de buracos em faces. Essa situação deverá ser contornada com a incorporação de superfícies curvas ao modelador.

VIII.3 – CONTRIBUIÇÕES DESTE TRABALHO

As contribuições obtidas durante o desenvolvimento deste trabalho estão relacionadas à aquisição de conhecimento, construção de software e produção científica.

Entre as contribuições referentes à aquisição de conhecimento, destacam-se:

– a obtenção de experiência no desenvolvimento de sistemas de modelagem de sólidos envolvendo as representações CSG e B-rep, além do instanciamento de primitivas e definição de objetos por varredura, criando uma base para futuros trabalhos e estudos nessa área, pelo GOPAC;

– a definição de uma nova estrutura de dados B-rep, que permite a operação de montagem sobre sólidos de variedade bidimensional, possibilitando a definição de sólidos manufaturáveis formados por vários componentes e a interpretação de maneira única da interface entre eles, o que garante a compatibilidade da malha de elementos finitos gerada. Essa estrutura supre uma necessidade específica da área de eletromagnetismo ainda não resolvida na literatura anterior;

– a definição de um padrão para estruturação do código e documentação, que poderá ser usado como base para futuros trabalhos do grupo. Além de permitir a obtenção de um produto mais homogêneo, essa padronização melhora o desenvolvimento do trabalho em equipe, uma vez que facilita o entendimento e o acesso a informações sobre o desenvolvimento e o produto;

– a consolidação da análise e programação orientadas para objetos na modelagem de sólidos visando a aplicações eletromagnéticas. Vale ressaltar que as primeiras experiências com a aplicação da orientação para objetos em eletromagnetismo ocorreram no início da década de 90 e estavam relacionadas à análise de equações diferenciais parciais por elementos finitos [For90, Zim92, Pèl92], sendo que nos trabalhos de Zimmermann e Pèlerin foi utilizada a linguagem *Smalltalk*, que não é eficiente do ponto de vista numérico. Aparentemente, a primeira aplicação desta metodologia na área de cálculo de campos eletromagnéticos tridimensionais foi feita pelo GOPAC [Sil93].

Entre as contribuições relacionadas à construção de software, destacam-se:

– o desenvolvimento de um projeto global para o pré-processador, incluindo não só o detalhamento da parte que foi implementada, mas também preparando o sistema para incorporar operações booleanas e superfícies curvas, parametrizar componentes e acoplar um gerador adaptativo de malha volumétrica;

– a implementação completa da nova estrutura de dados B-rep definida, o que permitiu verificar sua eficiência e validade prática;

– a implementação completa dos sub-sistemas de Interface e Representação, e a implementação parcial dos sub-sistemas de Geração de Malha – parte referente à geração da malha superficial – e Modelagem – restando implementar as operações booleanas –, o que deixa o modelador em um estágio que já permite sua utilização;

– a implementação do mecanismo para a definição e atribuição, aos componentes do modelo, das propriedades físicas e dos dados relacionados ao cálculo, bem como a implementação do mecanismo de geração do arquivo neutro. Apesar de já especificados [Gop96, Gop99], esses mecanismos ainda não haviam sido implementados em um sistema 3D;

– a obtenção de um código bem modularizado, possibilitando portar o modelador para outros ambientes com o mínimo possível de alterações – basicamente a interface;

– a padronização da documentação do sistema, obtida utilizando metodologias de análise e projeto orientados para objeto e normas internas de documentação para o código-fonte;

– o desenvolvimento de documentação *on-line* específica para o modelador, descrevendo detalhes de projeto, implementação e utilização pelo usuário, que deverão ser bastante úteis para futuras extensões a serem nele incluídas;

– a documentação de projeto e implementação do modelador bidimensional AutoPac, desenvolvido pelo GOPAC [Mag96a, Mes97].

Entre as contribuições relacionadas à produção científica, destacam-se:

– a participação na elaboração da nova versão do relatório "Definição da base de dados neutra para intercomunicação de dados em programas de cálculo de campos eletromagnéticos" [Gop99];

– a elaboração do artigo intitulado "*Solid Modeling Application in Electromagnetism*", publicado e apresentado no CBMag'96 [Mag96];

– a elaboração do artigo intitulado "*Requirements for a Solid Modeler Coupled to Finite-Element Mesh Generators*" [Mag97] e a participação na elaboração do artigo "*An Object-Oriented Platform for Teaching Finite Element Pre-Processor Programming and Design Techniques*" [Mes97], publicados e apresentados no COMPUMAG'97;

– a elaboração das versões estendidas dos artigos anteriores [Mag98a, Mes98], publicadas na revista *IEEE Transactions on Magnetics*;

– a elaboração do artigo intitulado "Tratamento de Fronteiras Internas em Modeladores de Sólidos Voltados para o Eletromagnetismo ", publicado e apresentado no CBMag'98 [Mag98b];

– a elaboração do artigo intitulado "*Exploring Inner Boundaries in Solid Modelers Applied to Electromagnetic Problems*" [Mag99c], publicado e apresentado no COMPUMAG'99;

– a elaboração da versão estendida do artigo anterior, submetida e aprovada para publicação na revista *IEEE Transactions on Magnetics* [Mag00];

– a elaboração dos artigos intitulados "Uma Estratégia para Gerar Malha de Elementos Finitos em Sólidos Definidos por Varredura" [Mag99d] e "Um Modelador de Sólidos Voltado para Aplicações em Eletromagnetismo" [Mag99e], publicados e apresentados no XX CILAMCE - Congresso Ibero Latino-Americano de Métodos Computacionais em Engenharia;

– a elaboração dos artigos intitulados "Essa Tal 'Compugrafia' " [Mag98c], "Apresentação Bidimensional de Sólidos" [Mag99], "Representação Computacional de Sólidos" [Mag99a] e "Tratamento de Fronteiras Internas em Modeladores de Sólidos" [Mag99f], publicados no boletim informativo CADTEC Mídia – EE/UFMG;

– a elaboração do artigo intitulado "Construção da Fronteira em Sólidos Definidos por Varredura", submetido ao CBMag'2000 [Mag00a];

– a elaboração e apresentação da palestra "Um Modelador de Sólidos Orientado para Objetos: Conceitos, Aplicações, Estudo de Caso", proferida no Seminário "Engenharia de Software e Programação Orientada para Objetos" oferecido pelo CADTEC em 01/10/1999;

– a elaboração de um painel sobre este trabalho de doutoramento e de um resumo intitulado "Desenvolvimento de um Modelador de Sólidos Voltado para Aplicações em Eletromagnetismo", apresentados na I Semana da Pós-Graduação da Universidade Federal de Minas Gerais, realizada em Setembro/1999 [Mag99b]. O trabalho foi premiado como um dos melhores na Área de Engenharia apresentados no evento.

– a elaboração de um novo resumo, *abstract* e painel para a "II Semana da Pós-Graduação da Universidade Federal de Minas Gerais", a ser realizada em Setembro/2000.

Outra contribuição não diretamente relacionada ao desenvolvimento do modelador mas integrante do programa de doutoramento foi a monitoria da disciplina "Programação Orientada para Objetos", durante a qual foi ministrado um curso de linguagem C via Internet para mais de duzentos alunos de todo o Brasil. Em duas edições subseqüentes do curso atuou como monitora, assessorando cerca de mil alunos inscritos em cada edição.

VIII.4 – SUGESTÕES PARA FUTUROS TRABALHOS

Existem diversos aperfeiçoamentos que podem ser incluídos neste modelador e que fogem ao escopo deste trabalho, entre eles:

– a incorporação de recursos já projetados e em desenvolvimento, como é o caso das operações booleanas [Nun00] e do acoplamento de um gerador de malhas volumétricas [Gon00];

– a incorporação de outros tipos de curva e superfície, aumentando o poder descritivo do modelador e resolvendo a questão da junção entre a malha e a representação por fronteira. Após a inclusão de superfícies curvas, será necessário definir e implementar uma forma de mapeamento das faces sobre as quais será gerada a malha. Uma possível estratégia consistiria em obter uma projeção planar da face, definida inicialmente por suas arestas e superfície curvas, sobre a qual ocorreria a geração da malha utilizando o módulo já implementado. Os nós e arestas da malha seriam então projetados de volta à face original, descrevendo assim a malha superficial;

– a implementação da varredura genérica, na qual a face ou sólido gerador pudesse percorrer uma trajetória qualquer e ainda sofrer transformações durante o percurso, o que poderia ser realizado utilizando grafos de controle de propagação [Mag87];

– a implementação de um mecanismo que permita desfazer operações errôneas ou indesejadas (*undo*) e refazer operações anteriormente desfeitas (*redo*) [Tor86];

– a adaptação do Subsistema de Interface do modelador para a plataforma Unix, utilizando provavelmente os recursos disponíveis no Kylix [Gar00];

– a aperfeiçoamento dos recursos de visualização, incluindo iluminação, efeitos de transparência e mapeamento de texturas sobre a superfície B-rep, além de mais recursos para animação;

– a implementação de operações locais de manipulação da forma sobre a estrutura B-rep, como arredondamento, chanframento e *tweaking*. Deve-se sempre ter em mente a necessidade de manter a compatibilidade entre as representações CSG e B-rep: qualquer alteração na representação B-rep deve ser refletida na estrutura CSG para que elas permaneçam compatíveis entre si e representando o mesmo componente sólido.

Também existem diversos trabalhos voltados para aplicações eletromagnéticas que podem ser acoplados a este modelador, uma vez que ele deve comportar-se como um pré-processador para várias aplicações. Entre esses trabalhos, destaca-se o desenvolvimento de um sistema de otimização da forma de dispositivos eletromagnéticos, englobando possivelmente as seguintes etapas: a forma do modelo inicial seria esboçada em relação a alguns parâmetros; a seguir, o modelo seria submetido à simulação,

ou seja, processado de forma estática ou dinâmica a partir da malha de elementos finitos gerada sobre ele; algoritmos de otimização poderiam tratar os resultados obtidos na simulação, propondo novos valores para os parâmetros de definição da forma do modelo inicial. Após sucessivas iterações do ciclo definição da forma / geração da malha / processamento / otimização, a geometria parametrizada do modelo poderia atingir uma solução ótima, melhorando a qualidade do modelo como um todo sem necessidade de muita intervenção humana.

Aproveitando a experiência adquirida não só com a representação de modelos sólidos adequados para o eletromagnetismo, mas também com o fornecimento e armazenamento de dados complementares para a análise eletromagnética, sugere-se que o GOPAC participe de maneira efetiva da proposta de normatização da base de dados para os problemas eletromagnéticos que está sendo discutida pelo comitê ISO STEP.

"Assuma consigo mesmo o compromisso
de estar sempre melhorando."
H. Jackson Brown Jr

APÊNDICE I – PADRONIZAÇÃO ADOTADA DURANTE O DESENVOLVIMENTO DO MODELADOR

AI.1 – A DOCUMENTAÇÃO DE MÓDULOS

A padronização de código adotada durante o desenvolvimento deste modelador baseia-se nas regras de programação para módulos em C/C++ estabelecidas por Staa [Sta98]. Cada módulo é composto por um módulo de definição (.h) e um módulo de implementação (.cpp). Os esqueletos dos módulos de definição e de implementação são apresentados nos quadros AI.1 e AI.2.

```
      ///////////////////////////////////////////////
(1)   // comentário padrão identificando o arquivo //
      ///////////////////////////////////////////////
(2)   //     cabeçalho do módulo de definição      //
      ///////////////////////////////////////////////

      // controle de definição do módulo
(3)   #if !defined ( nomeArq_MOD)
(3)      #define nomeArq_MOD

      // *** início do corpo do arquivo ***

      // controle de escopo do arquivo de definição
(4)      #if defined     ( nomeArq_OWN)
(4a)     #define  nomeArq_EXT
         // inclusões requeridas pelas bibliotecas do compilador
(5)          #include <biblio.h>
         // inclusões dos cabeçalhos dos módulos servidores requeridos
(6)          #include "servidores.h"
         // inclusões dos arquivos de tabelas de constantes requeridas
(7)          #include "tabelas.h"
      #else
(4b)     #define nomeArq_EXT extern
         // inclusões requeridas pelas bibliotecas do compilador
(5)          #include <biblio.h>
         // inclusões dos cabeçalhos dos módulos servidores requeridos
(6)          #include "servidores.h"
         // inclusões dos arquivos de tabelas de constantes requeridas
(7)          #include "tabelas.h"
      #endif

      // estruturas de dados exportadas pelo módulo
(8)      nomeArq_EXT declaração_da_variável_global_externa
(8a)  #if defined  ( nomeArq_EXT)
 .       = inicialização ; // não esquecer o ;
 .    #else
 .       ;                  // não esquecer o ;
(8a)  #endif


      // *** incluir aqui o corpo do arquivo ***


(4c)  #undef nomeArq_EXT

      // *** fim do corpo do arquivo ***

(3)   #endif
      ////////////////////////////////////////////////////
(9)   // comentário padrão identificando fim de arquivo //
      ////////////////////////////////////////////////////
```

Quadro AI.1 – Esqueleto do módulo de definição

Os módulos de definição (.h) são organizados da seguinte forma:

1. cabeçalho inicial do arquivo, contendo informações gerenciais;
2. cabeçalho do módulo, contendo informações técnicas;
3. inclusão do controle de definição do módulo, evitando sua inclusão duplicada;
4. controle de escopo vinculado às estruturas exportadas pelo módulo;
5. inclusões requeridas pelas bibliotecas do compilador;
6. inclusão de arquivo-cabeçalhos dos módulos servidores requeridos;
7. inclusões dos arquivos de tabelas de constantes requeridas;
8. declaração e possível definição das variáveis globais;
9. comentário identificando final do arquivo.

```
      ///////////////////////////////////////////////
   (1)  // comentário padrão identificando o arquivo //
      ///////////////////////////////////////////////
   (2)  //   cabeçalho do módulo de implementação    //
      ///////////////////////////////////////////////

      // inclusão do módulo de definição próprio
   (3)  #define     nomeArq_OWN
   (3)  #include  "nomeArq.h"
   (3)  #undef nomeArq_OWN

        // *** incluir aqui o corpo do arquivo ***
   (4)     // métodos devem ser colocados por classe e em ordem alfabética

      ////////////////////////////////////////////////////
   (5)  // comentário padrão identificando fim de arquivo //
      ////////////////////////////////////////////////////
```

Quadro AI.2 – Esqueleto do módulo de implementação

Os módulos de implementação são organizados da seguinte forma:

1. cabeçalho inicial do arquivo, contendo informações gerenciais;
2. cabeçalho do módulo, contendo informações técnicas;
3. inclusão do módulo de definição próprio;
4. métodos e funções alfabeticamente ordenadas por classe;
5. comentário identificando final do módulo.

Alguns comentários relacionados às estruturas apresentadas tornam-se relevantes:

– Deve-se evitar a inclusão desnecessária de arquivos, pois estes podem conter declarações que levam a conflitos ou a um consumo inútil de memória reservada para variáveis nunca utilizadas.

– Os módulos de definição e arquivos de tabela possuem controles para evitar sua inclusão duplicada ao compilar um módulo, como mostrado no item 3. O nome especificado será definido quando da primeira inclusão e sua definição não será excluída durante a compilação. Ele é composto pelo nome do arquivo seguido da seqüência _MOD, de forma a assegurar sua unicidade.

– No módulo de implementação, o código de inclusão do arquivo-cabeçalho está envolvido pela declaração nomeArq_OWN, como mostrado no item 3. Isto permite sinalizar quando o módulo de definição está sendo compilado junto com o de implementação e diferenciar o tratamento do módulo de definição como servidor.

– No módulo de definição o tratamento das inclusões são realizadas em separado para os casos em que o módulo está sendo compilado junto com o de implementação ou atuando como servidor para outros módulos (itens 5, 6 e 7). Isto garante a seqüencialização correta de inclusões de biblioteca, de módulos servidores e de arquivos de tabela e de dados.

– O uso da declaração nomeArq_OWN sinalizando a compilação do módulo de implementação permite controlar o escopo do arquivo de definição, como mostram os itens 4 e 8 do quadro AI.1: a declaração do nome terminado em _EXT controla a presença do declarador *extern*, que é gerado defronte de cada variável global se, e somente se, o módulo de definição for incluído para compilar um módulo cliente (4a, 4b). A presença do declarador *extern* faz com que a variável seja somente declarada, e sua ausência permite declarar, definir e ainda inicializar, como mostram os itens 8 e 8a do quadro AI.1. Ao final do código do módulo de definição deve-se retirar a definição de nomeArq_EXT (4c), assegurando que não ocorrerá interferência ao incluir módulos de definição dentro de outros módulos de definição.

– A interface de um módulo deve ser a menor possível. Portanto, módulos de definição devem conter somente as declarações de tipos, de constantes, de variáveis, de classes, de protótipos de funções e de código *inline* que efetivamente precisem ser externados para módulos cliente. Todas as demais declarações e código executável devem ser encapsulados e estar contidos exclusivamente no módulo de implementação. Este, por sua vez, não deve declarar nenhuma variável ou função global externa. Para assegurar estes requisitos, todas as declarações de variáveis globais encapsuladas e de protótipos de funções encapsuladas devem ser precedidas do declarador *static*.

– Outras recomendações envolvem facilidades para depuração do código e programação. Por exemplo, uma constante para depuração ou gravação pode ser definida e utilizada juntamente com comandos *ifdef's* ou *ifndef's* para isolar as partes do código que devem ser executadas somente quando o modo de depuração ou gravação estiver ativo, como mostra o quadro AI.3. O código isolado pode variar desde a inclusão de arquivos-cabeçalhos até o código de rotinas com lista variável de argumentos.

Módulo de Definição	Módulo de Definição
	//#define RECORD retirar o comentário para gravar
#define DEBUG	#ifndef RECORD
	#define REC(Y) Form->Memo->Lines->Add(Y) //mostra na tela
	//se não quiser ver nada, colocar só #define REC(Y)
	#else
	#define REC(Y) outfile<<(Y)<<endl //manda p/arquivo outfile
	# endif
Módulo de Implementação	Módulo de Implementação
	//abrir arquivo antes, se quiser gravar
#ifdef DEBUG	ofstream outfile "nomeArq";
//código isolado	//onde desejar
#endif	REC("conteúdo a ser opcionalmente gravado");

Quadro AI.3 – Exemplos de uso de estruturas *ifdef's* e *ifndefs* para flexibilizar o programa

AI.2 – A DOCUMENTAÇÃO DE PROGRAMAS

A documentação padronizada para o código é formada basicamente por cabeçalhos predefinidos para arquivos, classes, métodos ou funções, e comentários para atributos, variáveis e definições de tipos de dados. Todos os arquivos do código devem iniciar-se com um conjunto de informações gerenciais e técnicas, envolvendo: identificação do arquivo, descrição de seu conteúdo e histórico de alterações, como apresentado no quadro AI.4. Um rodapé como o apresentado no quadro AI.5 deverá finalizar todos os arquivos do código.

```
//=============================================================================
// Grupo de Otimizacao e Projeto Assistidos por Computador
//
// Projeto : GOPAC Solid Modeler             Versao: V1.0
//
// Arquivo : <nomeArquivo>.<extensão>        Versao: V1.01
// Caminho : Modeler\...\...
//
// Autores C - criador O - orientador A - atualizador:
//
// Conteudo:
// <descricao do conteudo geral do arquivo >
//
//=============================================================================
// Historico (registros em ordem decrescente):
// Vx.yy Modificado em  --/--/--  por  <nomes>
//      Motivo: <problema resolvido>
//      Descricao: <alteracao realizada / componentes afetados>
// V1.00 Criado em  --/--/--  por  <nomes>
//=============================================================================
// Documentacao : <nomeArquivo.htm>
//=============================================================================
```

Quadro AI.4 – Cabeçalho para início de um arquivo

```
//=============================================================================
//                       FIM DO ARQUIVO <nomeArquivo>.<extensão>
//=============================================================================
```

Quadro AI.5 – Rodapé para finalização de um arquivo

As classes devem ser comentadas da seguinte forma:

```
//=============================================================================
// Classe: <nome abreviado> - <nome por extenso>
//
// Super Classe: <classe pai> - Sub Classes: <classes filhas>
//
// Descricao:
// <descricao dos objetivos e caracteristicas especiais da classe>
//
// Documentacao : <arquivo.html#classe nomeClasse>
//=============================================================================
```

Quadro AI.6 – Formato básico para o cabeçalho de uma classe

Os métodos pertencentes a classes, bem como as funções independentes, devem ser comentados no módulo de implementação da seguinte forma:

```
//===========================================================================
// Metodo / Funcao: <nome abreviado> - <nome por extenso>
// Assinatura: <assinatura completa da funcao>
//
// Descricao:
// <descricao dos objetivos, acao e caracteristicas da funcao>
// <se sobrecarga: diferencas deste metodo em relacao aos outros de mesmo nome>
//
// Parametros recebidos:
// <explicacao de todos os parametros da lista de parametros da funcao>
//
// Valor retornado:
// <explicacao do valor retornado, discriminando valores possiveis, se for o caso>
//
// Parametros alterados:
// <relacao e explicacao de alteracoes possiveis nos parametros da lista>
//
// Outras variaveis / arquivos envolvidos:
// <relacao e explicacao de variaveis externas, globais ou arquivos lidos/alterados>
//
// Documentacao : <arquivo.html#metodo nomeClasse:nomeMetodo>
//===========================================================================
```

Quadro AI.7 – Formato básico para o cabeçalho de um método ou função no arquivo ".cpp"

Métodos ou funções independentes e seus parâmetros devem ser comentados no módulo de definição da seguinte forma:

```
<tipo> <nomeMetodo> (   //<descricao do metodo>
<tipo> <parametro>,     //<descricao do parametro>
<tipo> <parametro>)     //<descricao do parametro>
{<conteudo, se for o caso>}
```

Quadro AI.8 – Fomato básico para documentar um método ou função no arquivo-cabeçalho

Os atributos, bem como as variáveis diversas, devem ser comentados da seguinte forma:

```
<tipo> <nomeAtributo> <inicializacao,se existir>;  //<descricao do atributo>
```

Quadro AI.9 – Formato básico para atributos de classe e variáveis diversas

Os tipos de dados (enumerados ou não) e estruturas devem ser comentados da seguinte forma:

```
typedef <tipoBasico> <tipoNovo>;  //<descricao>                        ou

<typedef> struct <tipoNovo> {     //<descricao da estrutura>
        <tipo> <campo1>;          //<descricao do campo1>
        ...
        <tipo> <campon>;          //<descricao do campon>
};                                                                     ou

enum <label> {          //<descricao do label>
        <valor1>,       //<descricao do valor1>
        ...
        <valorn>        //<descrição do valorn>
};
<label> <varEnumerada>;   //<descrição da variável enumerada>
```

Quadro AI.10 – Formato básico para tipos de dados e estruturas

AI.3 – A DOCUMENTAÇÃO GLOBAL DO MODELADOR

Como todas as pessoas envolvidas na construção e utilização do modelador são de alguma forma desenvolvedores, ao invés de fazer documentações separadas para projeto, implementação e usuário, optou-se por elaborar uma única documentação *on-line*, em linguagem HTML. Esta documentação envolve não só o código implementado, como mostrado no item anterior, mas também todas as informações do modelador como um todo, incluindo sub-sistemas, assuntos dentro de cada subsistema, bases de dados manipuladas, etc. Criou-se desta forma uma árvore de arquivos HTML documentando cada componente do modelador. Os gabaritos definidos para sistema, sub-sistema, assunto, módulo, arquivo de dados, classe, atributo, método, tipo de dados definidos pelo usuário, informações gerenciais de arquivo, menu, submenu e comandos, utilizados em toda a documentação, são apresentados a seguir.

<table>
<tr><td>SIGLA DO SISTEMA
GSM</td><td>NOME COMPLETO
GOPAC Solid Modeler</td><td>VERSÃO
V1.0</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrever em linhas gerais o sistema></td></tr>
<tr><td colspan="3">ESTRUTURA FUNCIONAL
<apresentar uma visão geral do sistema, mostrando os principais componentes (subsistemas, arquivos, etc.) e como eles se relacionam – criar referências aos componentes></td></tr>
<tr><td colspan="3">REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas></td></tr>
<tr><td colspan="3">PRINCIPAIS CARACTERÍSTICAS
<relacionar características ligadas a equipamento, recursos utilizados, características de projeto e implementação></td></tr>
</table>

Quadro AI.11 – Gabarito para a documentação do sistema como um todo

<table>
<tr><td>SIGLA DO SUBSISTEMA
Sigla</td><td>NOME COMPLETO
Nome por extenso</td><td>VERSÃO
V1.0</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrever em linhas gerais o subsistema></td></tr>
<tr><td colspan="3">ESTRUTURA FUNCIONAL
<descrever a estrutura como um todo, se possível com uma ilustração>
Assunto: <nome do assunto, com referência ao gabarito de assunto>
    Módulos: <lista de módulos envolvidos, com referência ao gabarito do módulo>
Assunto: <nome do assunto com referência ao gabarito de assunto>
    Módulos: <lista de módulos envolvidos, com referência ao gabarito do módulo></td></tr>
<tr><td colspan="3">REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas></td></tr>
<tr><td colspan="3">PRINCIPAIS CARACTERÍSTICAS
<relacionar características ligadas a equipamento, recursos utilizados, características de projeto e implementação></td></tr>
</table>

Quadro AI.12 – Gabarito para a documentação de sub-sistemas

<table>
<tr><td>ASSUNTO
Nome abreviado</td><td>NOME COMPLETO
Nome por extenso</td><td>BIBLIOGRAFIA
[Aaa99], ...</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<explicar em linhas gerais o assunto e o envolvimento das classes com o assunto></td></tr>
<tr><td colspan="3">PRINCIPAIS CARACTERÍSTICAS, REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
< relacionar características de projeto e implementação>
<funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas></td></tr>
<tr><td colspan="3">DIAGRAMA ILUSTRATIVO
<figura do diagrama de classes></td></tr>
<tr><td colspan="3">CLASSES E MÓDULOS RELACIONADOS
Classe: <nome da classe, referenciando seu gabarito> - Módulo: <módulo relacionado, referenciando seu gabarito>
Classe: <nome da classe, referenciando seu gabarito> - Módulo: <módulo relacionado, referenciando seu gabarito>
...</td></tr>
</table>

Quadro AI.13 – Gabarito para a documentação de assuntos dentro de cada sub-sistema

<table>
<tr><td>MÓDULO
Nome do módulo</td><td>ARQUIVOS PERTENCENTES AO MÓDULO
Relação de arquivos ".cpp", ".h", ".dfm", etc., pertencentes ao módulo</td><td>PROJETO
GSM</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrição, identificando os objetivos do módulo e o que ele realmente faz></td></tr>
<tr><td colspan="3">MODO DE UTILIZAÇÃO
<descrever como proceder para incorporar o módulo a um programa e como o usuário interage com o módulo></td></tr>
<tr><td colspan="3">REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas></td></tr>
<tr><td colspan="3">AUTORES (C – criador, O – orientador, A – atualizador)
C – Nome do criadores
O – Nome do orientadores
A – Nome dos atualizadores</td></tr>
<tr><td colspan="3">HISTÓRICO
Vx.yy Modificado em --/--/-- por <nomes>
Motivo: <problema resolvido>
Descrição: <alteração realizada / componentes afetados>
Vx.yy Modificado em --/--/-- por <nomes>
Motivo: <problema resolvido>
Descrição: <alteração realizada / componentes afetados>
V1.00 Criado em --/--/-- por <nomes></td></tr>
<tr><td colspan="3">BIBLIOGRAFIA
<referências utilizadas para implementar o módulo, no formato [Aaa99] – página xx: conteúdo, com [Aaa99] remetendo a referência para sua descrição, no arquivo RefBiblio.htm></td></tr>
</table>

Quadro AI.14 – Gabarito para a documentação de módulos

<table>
<tr><td>ARQUIVO
Nome abreviado</td><td>NOME COMPLETO
Nome por extenso</td><td>BIBLIOGRAFIA
[Aaa99], ...</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrição, identificando os objetivos do arquivo e o que ele realmente faz></td></tr>
<tr><td colspan="3">PRINCIPAIS CARACTERÍSTICAS, REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<relacionar características de composição do arquivo – organização, tipo de acesso, etc>
<relacionar condições, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas></td></tr>
<tr><td colspan="3">ESPECIFICAÇÃO DE FORMATO
<para cada linha diferente do arquivo, mostrar os campos que definem o seu formato></td></tr>
<tr><td colspan="3">DESCRIÇÃO DOS CAMPOS RELACIONADOS
<para cada campo do formato, relacionar:>
<campo>: <tipo> <descrição> <domínio>
<campo>: <tipo> <descrição> <domínio>
...</td></tr>
</table>

Quadro AI.15 – Gabarito para a documentação de arquivos de dados a serem manipulados pelo sistema

ESTRUTURA Nome abreviado	NOME COMPLETO Nome por extenso	BIBLIOGRAFIA [Aaa99], ...
DESCRIÇÃO <explicar em linhas gerais a estrutura e o envolvimento das classes com a estrutura>		
PRINCIPAIS CARACTERÍSTICAS, REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS <relacionar características de projeto e implementação> <funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas>		
DIAGRAMA ILUSTRATIVO <figura do diagrama de classes>		
CLASSES E MÓDULOS RELACIONADOS Classe: <nome da classe, referenciando seu gabarito> - Módulo: <módulo relacionado, referenciando seu gabarito> Classe: <nome da classe, referenciando seu gabarito> - Módulo: <módulo relacionado, referenciando seu gabarito> ...		

Quadro AI.16 – Gabarito para a documentação de estruturas de dados

CLASSE Nome abreviado	NOME COMPLETO Nome por extenso	MÓDULO Módulo de origem
SUPER CLASSE Nome da classe pai	SUB CLASSES Nome das classes filhas	
DESCRIÇÃO <descrição dos objetivos e características especiais da classe>		
PARÂMETROS PARA A DEFINIÇÃO DA CLASSE <explicação de todos os parâmetros utilizados pelas construtoras da classe>		
INTERFACE – ATRIBUTOS PÚBLICOS <tipo> <atributo público – remeter para gabarito do atributo> : <significado> <tipo> <atributo público – remeter para gabarito do atributo > : <significado> ...		
INTERFACE – MÉTODOS PÚBLICOS <tipo> <método público– remeter para gabarito do método >(parâmetros) : <descrição sumária> <tipo> <método público– remeter para gabarito do método >(parâmetros) : <descrição sumária> ...		
PARTICULARIDADES - ATRIBUTOS PRIVADOS <tipo> <atributo público – remeter para gabarito do atributo> : <significado> <tipo> <atributo público – remeter para gabarito do atributo > : <significado> ...		
PARTICULARIDADES - MÉTODOS PRIVADOS <tipo> <método público– remeter para gabarito do método >(parâmetros) : <descrição sumária> <tipo> <método público– remeter para gabarito do método >(parâmetros) : <descrição sumária> ...		
PARTICULARIDADES - ESTRUTURAS DE DADOS <explicar características da estrutura de dados utilizada e como elas foram implementadas>		
REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS <funções, condições, atributos, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a liberdade de escolha das formas de construção envolvidas>		
BIBLIOGRAFIA <referências utilizadas para implementar a classe, no formato [Aaa99] – página xx: conteúdo, com [Aaa99] remetendo a referência para sua descrição, no arquivo RefBiblio.htm>		

Quadro AI.17 – Gabarito para a documentação de classes

ATRIBUTO Nome no código	NOME COMPLETO Nome completo, sem abreviações	CLASSE Classe onde está definido
DESCRIÇÃO <descrever o atributo e a sua unidade de medida, quando for o caso>		
CRITÉRIOS DE VALIDADE <descrever os valores válidos ou como é controlada sua validade>		
REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS <condições, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a implementação, a segurança, etc.>		

Quadro AI.18 – Gabarito para a documentação de atributos de classe

<table>
<tr><td>MÉTODO
Nome no código</td><td>NOME COMPLETO
Nome completo, sem abreviações</td><td>CLASSE
Classe onde está definido</td></tr>
<tr><td colspan="3">ASSINATURA / SOBRECARGA
<assinatura completa da função>
<diferenças deste método em relação aos outros de mesmo nome></td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrição dos objetivos, ação e características da função></td></tr>
<tr><td colspan="3">DADOS DE ENTRADA - PARÂMETROS
<tipo> <parâmetro> : <significado>
<tipo> <parâmetro> : <significado> ...
< explicação adicional dos parâmetros da lista acima, caso necessário></td></tr>
<tr><td colspan="3">DADOS DE ENTRADA – VARIÁVEIS GLOBAIS EXTERNAS OU PÚBLICAS, PRÓPRIAS DO MÓDULO
<tipo> <variável global – remeter para gabarito da variável global> : <significado>
<tipo> <variável pública – remeter para gabarito da variável pública> : <significado> ...</td></tr>
<tr><td colspan="3">DADOS DE ENTRADA – VARIÁVEIS GLOBAIS EXTERNAS OU PÚBLICAS, DE OUTROS MÓDULOS
<tipo> <variável global – remeter para gabarito da variável global> : <origem> <significado>
<tipo> <variável pública – remeter para gabarito da variável pública> : <origem> <significado> ...
<variáveis externas devem ser usadas com cautela, preferindo-se funções de acesso a dados encapsulados></td></tr>
<tr><td colspan="3">ARQUIVOS MANIPULADOS
<arquivo – remeter para gabarito do arquivo> : <acesso – leitura / gravação / ambos> <significado>
<arquivo – remeter para gabarito do arquivo> : <acesso – leitura / gravação / ambos> <significado> ...</td></tr>
<tr><td colspan="3">VALOR RETORNADO
< natureza do valor retornado, discriminar valores possíveis></td></tr>
<tr><td colspan="3">DADOS DE SAÍDA – PARÂMETROS ALTERADOS
<tipo> <parâmetro> : <significado>
<tipo> <parâmetro> : <significado> ...
< explicação adicional dos parâmetros da lista acima, caso necessário></td></tr>
<tr><td colspan="3">DADOS DE SAÍDA – VARIÁVEIS GLOBAIS EXTERNAS OU PÚBLICAS, PRÓPRIAS DO MÓDULO, QUE FORAM ALTERADAS
<tipo> <variável global – remeter para gabarito da variável global> : <significado>
<tipo> <variável pública – remeter para gabarito da variável pública> : <significado> ...</td></tr>
<tr><td colspan="3">DADOS DE SAÍDA – VARIÁVEIS GLOBAIS EXTERNAS OU PÚBLICAS, DE OUTROS MÓDULOS, QUE FORAM ALTERADAS
<tipo> <variável global – remeter para gabarito da variável global> : <origem> <significado>
<tipo> <variável pública – remeter para gabarito da variável pública> : <origem> <significado> ...
<variáveis externas devem ser usadas com cautela, preferindo-se funções de acesso a dados encapsulados></td></tr>
<tr><td colspan="3">REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<condições, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a implementação, a segurança, etc.></td></tr>
<tr><td colspan="3">BIBLIOGRAFIA
<referências utilizadas para implementar a classe, no formato [Aaa99] – página xx: conteúdo, com [Aaa99] remetendo a referência para sua descrição, no arquivo RefBiblio.htm></td></tr>
</table>

Quadro AI.19 – Gabarito para a documentação de métodos de classes e funções

<table>
<tr><td>TIPO
Nome no código</td><td>NOME COMPLETO
Nome completo, sem abreviações</td><td>SUBSISTEMA / MÓDULO / CLASSE
Contexto onde está definido</td></tr>
<tr><td colspan="3">DESCRIÇÃO
<descrever o tipo como um todo, sua lei de formação bem como detalhes de sua estrutura de dados></td></tr>
<tr><td colspan="3">CRITÉRIOS DE VALIDADE OU VALORES POSSÍVEIS PARA TIPOS ENUMERADOS
<valores definidos></td></tr>
<tr><td colspan="3">REQUISITOS, RESTRIÇÕES OU HIPÓTESES ASSUMIDAS
<condições, propriedades ou características: a serem satisfeitas ou que não devem ser satisfeitas; necessárias mas assumidas como satisfeitas antes do desenvolvimento; que restringem a implementação, a segurança, etc.></td></tr>
<tr><td colspan="3">BIBLIOGRAFIA
<referências utilizadas para implementar o tipo, no formato [Aaa99] – página xx: conteúdo, com [Aaa99] remetendo a referência para sua descrição, no arquivo RefBiblio.htm></td></tr>
</table>

Quadro AI.20 – Gabarito para a documentação de tipos de dados definidos pelo usuário

<table>
<tr><td>GESTOR
GOPAC</td><td colspan="2">PROJETO
GSM – GOPAC Solid Modeler</td><td>MÓDULO
Nome do Módulo</td></tr>
<tr><td>ARQUIVO
Nome do Arquivo</td><td>VERSÃO
Vx.yy</td><td colspan="2">CAMINHO RELATIVO AO DIRETÓRIO PRINCIPAL DO MODELADOR
Modeler\...\...</td></tr>
<tr><td colspan="4">CONTEÚDO
<descrição do conteúdo geral do arquivo - não fornecer dados específicos></td></tr>
</table>

Quadro AI.21 – Gabarito para a documentação de informações gerenciais sobre o arquivo

<table>
<tr><td>SUBMENU
Nome do Submenu</td><td>HIERARQUIA NO MENU
Menu / Submenu / ...</td><td>TOOLBAR CORRESPONDENTE
Nome do toolbar</td></tr>
<tr><td colspan="3">OBJETIVO
Hint curto | Hint longo <explicação resumida do submenu, colocada no campo Hint></td></tr>
<tr><td colspan="3">EXPLICAÇÃO DE SEU FUNCIONAMENTO
<descrição das características gerais de funcionamento, do ponto de vista do usuário – para que serve, como utilizar></td></tr>
<tr><td colspan="3">EXPLICAÇÃO DE SUA IMPLEMENTAÇÃO
<descrição resumida de como foi implementado – para submenu, geralmente colocar o descrito abaixo>
Nenhuma. Apenas definida na estrutura de menu.</td></tr>
</table>

Quadro AI.22 – Gabarito para a documentação de menu e sub-menus

<table>
<tr><td>ÍCONE
figura</td><td>SIGLA
NOME</td><td>CAMINHO MENU
Menu / Submenu / ...</td><td>TOOLBAR ONDE ESTÁ DEFINIDO
Nome do toolbar</td></tr>
<tr><td colspan="4">OBJETIVO
Hint curto | Hint longo <explicação resumida do comando, colocada no campo Hint></td></tr>
<tr><td colspan="4">EXPLICAÇÃO DE SEU FUNCIONAMENTO
<descrição das características gerais de funcionamento, do ponto de vista do usuário – para que serve, como utilizar></td></tr>
<tr><td colspan="4">AUTÔMATO PARA FORNECIMENTO DE PARÂMETROS
<uma das opções abaixo, dependendo de possuir parâmetros, recebê-los a partir da caixa de diálogo ou não possuir>
[<estado1> - Mensagem para entrada do parâmetro
<o que espera receber> - Mensagem exibida caso ocorra erro
<estado2> - Mensagem para entrada do parâmetro
<o que espera receber> - Mensagem exibida caso ocorra erro
...] ou
Nenhum. Parâmetros fornecidos pela caixa de diálogo. ou
Comando ainda não implementado.</td></tr>
<tr><td colspan="4">CAIXA DE DIÁLOGO ABERTA PELO COMANDO
<figura contendo a caixa de diálogo e alguma explicação adicional, se esta existir></td></tr>
<tr><td colspan="4">CONSISTÊNCIAS REALIZADAS E MENSAGENS EMITIDAS
<explicar consistência e mostrar mensagem emitida></td></tr>
<tr><td colspan="4">EXPLICAÇÃO DE SUA IMPLEMENTAÇÃO
<informações gerais sobre a implementação, que remetam para detalhes e arquivos em documentação específica ></td></tr>
</table>

Quadro AI.23 – Gabarito para a documentação de comandos implementados no modelador

"Não se pode ensinar tudo a alguém,
pode-se apenas ajudá-lo a encontrar por si mesmo."
Galileu Galilei

APÊNDICE II – COMANDOS DISPONÍVEIS

O quadro a seguir relaciona os comandos disponíveis no modelador, a serem digitados na linha de comando, selecionados pelo menu ou ativados pelo ícone existente na barra de ferramentas. A coluna SITUAÇÃO ATUAL indica a estado de desenvolvimento do comando.

NOME DO COMANDO: DESCRIÇÃO	POSIÇÃO NO MENU BARRA DE FERRAMENTA	SITUAÇÃO ATUAL	ÍCONE
NEW: Reinicializa o ambiente e cria um novo modelo com nome automático	File | New File	pronto	
OPEN: Lê um arquivo de modelo existente em disco e o carrega para a memória	File | Open File	pronto	
SAVE: Grava em disco um arquivo ao modelo em uso na memória	File | Save File	pronto	
SAVEAS: Grava em disco o modelo em uso com um outro nome e | ou caminho	File | Save As File	pronto	
NEUTRALFILE: Grava em disco o modelo em uso seguindo o formato neutro	File | Export | Neutral File File	pronto	
SAVEDXF: Grava em disco o modelo em uso seguindo o formato DXF	File | Export| DXF File	a desenvolver	
SAVETABLES: Grava em disco as tabelas com características físicas do modelo	File | Export | Tables File	pronto	
LOADTABLES: Lê arquivo contendo tabelas que definem características físicas	File | Import | Tables File	pronto	
BUILDBREP: Constrói um modelo B-rep a partir de um arquivo do tipo ".brp"	File | Import | B-rep Model File	pronto	
SAVEBMP: Grava em disco um arquivo ".bmp" com a visão atual do modelo	File | Save BMP File File	pronto	
PRINT: Imprime o modelo em uso como apresentado na área gráfica	File | Print File	pronto	
SETPRINTER: Define características relacionadas à impressão	File | Print Setup File	pronto	
EXIT: Finaliza a sessão do GOPAC Solid Modeler	File | Exit Não tem – fica no campo superior direito da janela principal	pronto	
COPY: Copia um ou mais elementos do modelo	Edit | Copy Edit	pronto	
ERASE: Exclui um ou mais elementos do modelo	Edit | Erase Edit	pronto	
ROTATE: Rotaciona um ou mais elementos do modelo	Edit | Rotate Edit	pronto	
MOVE: Desloca um ou mais elementos do modelo	Edit | Move Edit	pronto	
SCALE: Escala um ou mais elementos do modelo	Edit | Scale Edit	pronto	
MIRROR: Espelha um ou mais elementos do modelo	Edit | Mirror Edit	pronto	
GROUP: Agrupa elementos para facilitar manipulação	Edit | Group Edit	a desenvolver	
UNGROUP: Desagrupa elementos anteriormente agrupados	Edit | Ungroup Edit	a desenvolver	
COPY2D: Copia um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Copy Edit2D	pronto	
ERASE2D: Exclui um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Erase Edit2D	pronto	
ROTATE2D: Rotaciona um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Rotate Edit2D	pronto	
MOVE2D: Desloca um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Move Edit2D	pronto	
SCALE2D: Escala um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Scale Edit2D	pronto	
MIRROR2D: Espelha um ou mais primitivas 2D (perfis) do modelo	Edit | 2D Primitive | Mirror Edit2D	pronto	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas

Nome do Comando: Descrição	Posição no Menu Barra de Ferramenta	Situação Atual	Ícone
MERGESIL: Une duas silhuetas abertas, incorporando a segunda na primeira	Edit | Silhouette | Merge Silhouettes Edit2D	pronto	
OPENSIL: Remove ligação entre os pontos inicial e final de uma silhueta fechada	Edit | Silhouette | Open Silhouette Edit2D	pronto	
CLOSESIL: Une os pontos inicial e final de uma silhueta aberta	Edit | Silhouette | Close Silhouette Edit2D	pronto	
CIRCLE: Enumera as opções para definir um círculo	Create | 2DPrimitive | Circle 2D Primitives	pronto	
CIRCLECR: Gera um círculo pelo centro e raio	Create | 2DPrimitive | Circle | Center,Radius 2D Primitives	pronto	
CIRCLECD: Gera um círculo pelo centro e diâmetro	Create | 2DPrimitive | Circle | Center,Diameter 2D Primitives	pronto	
CIRCLE2P: Gera círculo por 2 pontos diametralmente opostos	Create | 2DPrimitive | Circle | 2 Opposite Points 2D Primitives	pronto	
CIRCLE3P: Gera círculo que passa pelos 3 pontos definidos	Create | 2DPrimitive | Circle | 3 Points 2D Primitives	pronto	
CIRCLETTR: Gera círculo tangente a 2 elementos com raio definido	Create | 2DPrimitive | Circle | Tan,Tan,Radius 2D Primitives	a desenvolver	
ELLIPSE: Gera elipse pelo centro e raios horizontal e vertical	Create | 2DPrimitive | Ellipse 2D Primitives	pronto	
POLYGON: Enumera as opções para definir um polígono	Create | 2DPrimitive | Polygon | Rectangle 2D Primitives	pronto	
POLY2P: Gera retângulo por 2 pontos extremos opostos	Create | 2DPrimitive | Polygon | Rectangle 2D Primitives	pronto	
POLYCLV: Gera polígono regular pelo centro, tamanho lado e nr vértices	Create | 2Dprimitive | Polygon | Regular | Center, Length, #Edges 2D Primitives	pronto	
POLYINSC: Gera polígono regular inscrito pelo centro, raio e nr vértices	Create | 2DPrimitive | Polygon | Regular | Inscribe: Center, Radius, #Edges 2D Primitives	pronto	
POLYCIRC: Gera polígono regular circunscrito pelo centro, raio e nr vértices	Create | 2DPrimitive | Polygon | Regular | Circunsc: Center, Rad, #Edges 2D Primitives	pronto	
POLYVERT: Gera polígono definido pelos seus vértices (mínimo 3)	Create | 2DPrimitive | Polygon | Vertexes 2D Primitives	pronto	
POLYFILE: Gera polígono definido a partir de arquivo ".pol"	Create | 2DPrimitive | Polygon | ".pol" File 2D Primitives	pronto	
SILHOUETTE: Gera silhueta com linhas, arcos circulares e elípticos	Create | 2DPrimitive | Silhouette 2D Primitives	pronto	
ARC3P: Gera um arco definido por 3 pontos nele contidos – inicial, qualquer, final	Create | 2DPrimitive | Silhouette | Arc: 3 Points 2D Primitives	pronto	
ARCSCE: Gera um arco definido pelo início, centro e fim	Create | 2DPrimitive | Silhouette | Arc: Start, Center, End 2D Primitives	pronto	
ARCSCA: Gera um arco definido pelo início, centro e ângulo	Create | 2DPrimitive | Silhouette | Arc: Start, Center, Angle 2D Primitives	pronto	
ARCSCL: Gera um arco definido pelo início, centro e comprimento da corda	Create | 2DPrimitive | Silhouette | Arc: Start, Center, Length 2D Primitives	pronto	
ARCSEA: Gera um arco definido pelo início, fim e ângulo	Create | 2DPrimitive | Silhouette | Arc: Start, End, Angle 2D Primitives	pronto	
ARCSED: : Gera um arco definido pelo início, fim e direção	Create | 2DPrimitive | Silhouette | Arc: Start, End, Direction 2D Primitives	a desenvolver	
ARCSER: : Gera um arco definido pelo início, fim e raio	Create | 2DPrimitive | Silhouette | Arc: Start, End, Radius 2D Primitives	pronto	
PLINE: Gera uma polilinha definida por uma seqüêcia de vértices	Create | 2DPrimitive | Silhouette | Polyline 2D Primitives	pronto	
COMPOUNDSIL: Agrupa um conjunto de silhuetas, formando perfis vazados	Create | 2DPrimitive | Compound Silhouette 2D Primitives	pronto	
RADIALSIL: Repete uma silhueta aberta de forma radial	Create | 2DPrimitive | Radial Silhouette 2D Primitives	pronto	
BLOCK: Gera um bloco de base retangular e deslocamento livre	Create | 3DPrimitive | Block 3D Primitives	pronto	
SPHERE: Gera uma esfera a partir do centro e raio	Create | 3DPrimitive | Sphere 3D Primitives	pronto	
HSPHERE: Gera um hemisfério pelo centro, raio e posição	Create | 3DPrimitive | Hemisphere 3D Primitives	pronto	
ELLIPSOID: Gera um elipsóide pelo centro e raios em X, Y e Z	Create | 3DPrimitive | Ellipsoid 3D Primitives	pronto	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas (continuação)

NOME DO COMANDO: DESCRIÇÃO	POSIÇÃO NO MENU BARRA DE FERRAMENTA	SITUAÇÃO ATUAL	ÍCONE
CYLINDER: Gera um cilindro circular ou elíptico com deslocamento livre	Create | 3DPrimitive | Cylinder 3D Primitives	pronto	
CYLINDERCR: Gera um cilindro circular com base definida pelo centro e raio	Create | 3DPrimitive | Cylinder | Center, Radius, Displ. 3D Primitives	pronto	
CYLINDERCD: Gera um cilindro circular com base pelo centro e diâmetro	Create | 3DPrimitive | Cylinder | Center, Diameter, Displ. 3D Primitives	pronto	
CYLINDER2P: Gera um cilindro circular com base por 2 pontos opostos	Create | 3DPrimitive | Cylinder | 2 Opposite Points, Displ. 3D Primitives	pronto	
CYLINDER3P: Gera um cilindro circular com base definida por 3 pontos	Create | 3DPrimitive | Cylinder | 3 Points, Displ. 3D Primitives	pronto	
CYLINDERTTR: Gera um cilindro circular ou com base definida por tan,tan,raio	Create | 3DPrimitive | Cylinder | Tan, Tan, Radius, Displ. 3D Primitives	a desenvolver	
CYLINDEREL: Gera um cilindro elíptico com deslocamento livre	Create | 3DPrimitive | Cylinder | Center, Rx, Ry, Displ. 3D Primitives	pronto	
CONE: Gera um cone ou tronco circular ou elíptico com deslocamento livre	Create | 3DPrimitive | Cone 3D Primitives	pronto	
CONECR: Gera um cone circular com base definida pelo centro e raio	Create | 3DPrimitive | Cone | Center, Radius, Displ, Height 3D Primitives	pronto	
CONECD: Gera um cone circular com base pelo centro e diâmetro	Create | 3DPrimitive | Cone | Center, Diameter, Displ, Height 3D Primitives	pronto	
CONE2P: Gera um cone circular com base por 2 pontos opostos	Create | 3DPrimitive | Cone|2Opposite Points, Displ, Height 3D Primitives	pronto	
CONE3P: Gera um cone circular com base definida por 3 pontos	Create | 3DPrimitive | Cone | 3 Points, Displ, Height 3D Primitives	pronto	
CONETTR: Gera um cone circular ou com base definida por tan,tan,raio	Create | 3DPrimitive | Cone | Tan, Tan, Radius, Height 3D Primitives	a desenvolver	
CONEEL: Gera um cone elíptico com deslocamento livre	Create | 3DPrimitive | Cone | Center, Rx, Ry, Height 3D Primitives	pronto	
PRISM: Gera um prisma com base poliedral e deslocamento livre	Create | 3DPrimitive | Prism 3D Primitives	pronto	
PRISM2P: Gera um prisma com base retangularl e deslocamento livre	Create | 3DPrimitive | Prism | Rectangle, Displ. 3D Primitives	pronto	
PRISMCLV: Gera um prisma com base poliedral pelo centro, #arestas, lado	Create | 3DPrimitive | Prism | RegBase, Displ | Center ,#Edges, Length 3D Primitives	pronto	
PRISMINSC: Gera um prisma com base poliedral regular inscrita	Create | 3DPrimitive | Prism | RegBase, Displ | Inscr:Center,Radius,#Edges 3D Primitives	pronto	
PRISMCIRC: Gera um prisma com base poliedral regular circunscrita	Create | 3DPrimitive | Prism | RegBase, Displ | Circ:Center,Radius,#Edges 3D Primitives	pronto	
PRISMVERT: Gera um prisma com base poliedral definida pelos vértices	Create | 3DPrimitive | Prism | Vertexes, Displ. 3D Primitives	pronto	
PRISMFILE: Gera um prisma com base poliedral definida em arquivo ".pol"	Create | 3DPrimitive | Prism | ".pol" File, Displ 3D Primitives	pronto	
PYRAMID: Gera uma pirâmide ou tronco com base poligonal e desloc. livre	Create | 3DPrimitive | Pyramid 3D Primitives	pronto	
PYRAMID2P: Gera um pirâmide com base retangularl e deslocamento livre	Create | 3DPrimitive | Pyramid | Rectangle, Displ, Height 3D Primitives	pronto	
PYRAMIDCLV: Gera um pirâmide com base poliedral pelo centro, #arestas, lado	Create | 3DPrimitive | Pyramid | RegBase, Displ, Height | Cen, #Edg, Len 3D Primitives	pronto	
PYRAMIDINSC: Gera um pirâmide com base poliedral regular inscrita	Create | 3DPrimitive | Pyramid | RegBase,Displ,Height|Insc:Cen,Rad,#Edg 3D Primitives	pronto	
PYRAMIDCIRC: Gera um pirâmide com base poliedral regular circunscrita	Create | 3DPrimitive | Pyramid | RegBase,Displ,Height|Circ:Cen,Rad,#Edg 3D Primitives	pronto	
PYRAMIDVERT: Gera um pirâmide com base poliedral definida pelos vértices	Create | 3DPrimitive | Pyramid | Vertexes, Displ, Height. 3D Primitives	pronto	
PYRAMIDFILE: Gera um pirâmide com base poliedral definida em arquivo "pol"	Create | 3DPrimitive | Pyramid | ".pol" File, Displ, Height 3D Primitives	pronto	
TORUS: Gera um toro pelo centro e raios externo e interno	Create | 3DPrimitive | Torus 3D Primitives	pronto	
ROTSWEEP: Realiza a varredura rotacional de um perfil aberto ou fechado	Create | Sweep Primitive | SimpleRotation Sweep	pronto	
TRANSWEEP: Realiza a varredura translacional simples de um perfil fechado	Create | Sweep Primitive | SimpleTranslation Sweep	pronto	
CONICSWEEP: Realiza a varredura translacional cônica de um perfil fechado	Create | Sweep Primitive | ConicTranslation Sweep	pronto	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas (continuação)

Nome do Comando: Descrição	Posição no Menu Barra de Ferramenta	Situação Atual	Ícone
UNION: Realiza a união de dois componentes 3D do modelo	Create | CSG Tree | Union CSG	a desenvolver	
INTERSECTION: Realiza a interseção de dois componentes 3D do modelo	Create | CSG Tree | Intersection CSG	a desenvolver	
DIFERENCE: Realiza a diferença (A-B) de dois componentes 3D do modelo	Create | CSG Tree | Difference CSG	a desenvolver	
ASSEMBLE: Realiza a montagem de dois componentes 3D do modelo	Create | CSG Tree | Assemble CSG	parcial	
NEWLABEL: Define novos rótulos com características físicas para componentes	Label | NewLabel Labe	pronto	
EDITLABEL: Define novos dados para os rótulos com características físicas	Label | Edit Label Label	pronto	
LABELVERTEX: Atribui um label (rótulo) geometria / malha a um ou mais vértices	Label | Vertexl Label	pronto	
LABELEDGE: Atribui um label (rótulo) de geometria / malha a uma ou mais arestas	Label | Edge Label	pronto	
LABELFACE: Atribui um label (rótulo) de geometria / malha a uma ou mais faces	Label | Face Label	pronto	
LABELREGION: Atribui um label (rótulo) geometria /malha a uma ou mais regiões	Label | Region Label	pronto	
EDITTABLES: Define novos dados para as tabelas de características físicas	Label | Edit Tables Label	pronto	
SELECTWP: Define um novo plano de trabalho	Properties | WorkPlane | SelectWorkPlane Work Plane	pronto	
MOVEWP: Move o plano de trabalho atual sobre o plano infinito que o contém	Properties | WorkPlane | MoveWorkPlane Work Plane	pronto	
SETVIEWP: Reposiciona o plano de trabalho para a posição atual da câmera	Properties | WorkPlane | SetViewportAsWorkPlane Work Plane	pronto	
SETFACEWP: Reposiciona o plano de trabalho, centrando-o com uma face	Properties | WorkPlane | SetFaceAsWorkPlane Work Plane	pronto	
CHANGECOLORS: Altera cores de uma região já gerada	Properties | ChangeDrawingColors Properties	pronto	
PREFERENCES: Define preferêcnias de desenho e configuração do modelo	Properties | Preferences Properties	pronto	
ZWINDOW: Executa o zoom na área definida	View | Zoom | Window Zoom	pronto	
ZIN: Avança para dentro do modelo, ampliando sua visão	View | Zoom | In Zoom	pronto	
ZOUT: Retrocede para fora do modelo, encolhendo sua visão	View | Zoom | Out Zoom	pronto	
ZALL: Mostra todo o espaço de trabalho definido	View | Zoom | All Zoom	pronto	
ZPREVIOUS: Volta para a visão definida anteriormente	View | Zoom | Previous Zoom	pronto	
VFRONT: Posiciona a tela na frente do cubo de visão	View | SelectViewport | Front View	pronto	
VBACK: Posiciona a tela no fundo do cubo de visão	View | SelectViewport | Back View	pronto	
VTOP: Posiciona a tela em cima do cubo de visão	View | SelectViewport | Top View	pronto	
VBOTTOM: Posiciona a tela embaixo do cubo de visão	View | SelectViewport | Bottom View	pronto	
VLEFT: Posiciona a tela à esquerda do cubo de visão	View | SelectViewport | Left View	pronto	
VRIGHT: Posiciona a tela à direita do cubo de visão	View | SelectViewport | Right View	pronto	
VBACKRIGHT: Posiciona a tela no fundo e à direita do cubo de visão	View | SelectViewport | BackRight View	pronto	
VBACKLEFT: Posiciona a tela no fundo e à esquerda do cubo de visão	View | SelectViewport | BackLeft View	pronto	
VFRONTRIGHT: Posiciona a tela na frente e à direita do cubo de visão	View | SelectViewport | FrontRight View	pronto	
VFRONTLEFT: Posiciona a tela na frente e à esquerda do cubo de visão	View | SelectViewport | FrontLeft View	pronto	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas (continuação)

NOME DO COMANDO: DESCRIÇÃO	POSIÇÃO NO MENU BARRA DE FERRAMENTA	SITUAÇÃO ATUAL	ÍCONE
ROTCAMX: Rotação da câmera segundo o eixo X	View | CameraRotation | AroundX CameraRotation	pronto	
ROTCAMY: Rotação da câmera segundo o eixo Y	View | CameraRotation | AroundY CameraRotation	pronto	
ROTCAMZ: Rotação da câmera segundo o eixo Z	View | CameraRotation | AroundZ CameraRotation	pronto	
ROTCAMAUTO: Rotação automática da câmera segundo os eixos X, Y e Z	View | CameraRotation | AutomaticRotation CameraRotation	pronto	
MOVCAMIN: Penetra com a câmera para dentro do modelo	View | CameraMoving | In CameraMoving	pronto	
MOVCAMOUT: Afasta a câmera para fora do modelo	View | CameraMoving | Out CameraMoving	pronto	
MOVCAMUP: Desloca a câmera para cima, abaixando o modelo	View | CameraMoving | Up CameraMoving	pronto	
MOVCAMDOWN: Desloca a câmera para baixo, levantando o modelo	View | CameraMoving | Down CameraMoving	pronto	
MOVCAMRIGHT: Desloca a câmera para a direita, movendo o modelo p/ esquerda	View | CameraMoving | Right CameraMoving	pronto	
MOVCAMLEFT: Desloca a câmera para a esquerda, movendo o modelo p/ direita	View | CameraMoving | Left CameraMoving	pronto	
BKPLANECLOSER: Desloca plano de fundo para frente, diminuindo o alcance	View | CameraMoving | BringBackPlaneCloser CameraMoving	pronto	
BKPLANEFARTHER: Desloca plano de fundo para trás, aumentando o alcance	View | CameraMoving | BringBackPlaneFarther CameraMoving	pronto	
MOVCAMAUTO: Movimento automático da câmera entrando e saindo do modelo	View | CameraMoving | AutomaticMoving CameraMoving	pronto	
HOLDCAM: Armazena a posição atual da câmera para futuro reposisionamento	View | CameraMoving | HoldCamera CameraMoving	pronto	
FETCHCAM: Volta a câmera para a posição anteriormente armazenada	View | CameraMoving | FetchCamera CameraMoving	pronto	
STEREO: Apresenta uma visão estéreo das arestas modelo	View | CameraMoving | StereoEdges Properties	pronto	
HELP: Apresenta uma ajuda *on-line* sobre os comandos e recursos do modelador	Assist | Help Assist	parcial	
ABOUT: Acessa informações na rede sobre o desenvolvimento do modelador	Assist | About Assist	pronto	
HOLDMODEL: Armazena a situação atual do modelo para futura recuperação	Assist | HoldModel Assist	a desenvolver	
FETCHMODEL: Restaura a situação do modelo anteriormente armazenada	Assist | FetchModel Assist	a desenvolver	
ENDPOINT: Obtém ponto final do segmento de uma primitiva (comando auxiliar)	Menu pop-up – End Segment Point Object Snap	pronto	
ENDSILPOINT: Obtém ponto final de uma primitiva (comando auxiliar)	Menu pop-up – End Silhouette Point Object Snap	pronto	
MIDPOINT:Obtém ponto médio do segmento de uma primitiva (comando auxiliar)	Menu pop-up – Middle Point Object Snap	pronto	
CENTERPOINT: Obtém o centro de uma primitiva (comando auxiliar)	Menu pop-up – Center Point Object Snap	pronto	
BARICPOINT: Obtém o baricentro de uma primitiva (comando auxiliar)	Menu pop-up – Baricenter Point Object Snap	pronto	
INTERSECTPOINT: Obtém a interseção entre 2 primitivas (comando auxiliar)	Menu pop-up – Intersection Point Object Snap	pronto	
VERTEXPOINT: Obtém o vértice mais próximo das primitivas (comando auxiliar)	Menu pop-up – Vertex Point Object Snap	pronto	
NEARESTPOINT: Obtém o ponto mais próximo das primitivas (comando auxiliar)	Menu pop-up – Nearest Point Object Snap	pronto	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas (continuação)

NOME DO COMANDO: DESCRIÇÃO	POSIÇÃO NO MENU BARRA DE FERRAMENTA	SITUAÇÃO ATUAL	ÍCONE
COORDS: Obtém as coordenadas X,Y,Z do ponto selecionado	Assist | Inquiry | Coords Inquiry	pronto	
DISTANCE: Obtém a distância entre os pontos selecionados	Assist | Inquiry | Distance Inquiry	pronto	
ANGLE: Obtém o ângulo compreendido entre 3 pontos selecionados	Assist | Inquiry | Angle Inquiry	pronto	
VECTOR: Obtém o deslocamento em X,Y,Z entre 2 pontos selecionados	Assist | Inquiry | Vector Inquiry	pronto	
PERIMETER: Obtém o comprimento total do contorno de uma primitiva 2D	Assist | Inquiry | Perimeter Inquiry	pronto	
SETTINGS: Obtém características gerais utilizadas no modelo	Assist | Inquiry | Settings Inquiry	a desenvolver	

Quadro AII.1 – Comandos disponíveis: nome, posição no menu e ícone na barra de ferramentas (continuação)

"Mestre não é quem sempre ensina,
mas quem, de repente, aprende"
Guimarães Rosa

REFERÊNCIAS BIBLIOGRÁFICAS

[Ala90] ALAGAR, V. S., BUI, T. D. e PERIYASAMY, K. Semantic CSG trees for finite element analysis. Computer-aided design, Guildford, v. 22, n. 4, p. 194-198, may 1990.

[Ale61] ALEXANDROFF, P. Elementary concepts of topology. New York: Dover, 1961.

[Arb90] ARBAB, F. Set models and boolean operations for solids and assemblies. IEEE computer graphics and applications, Los Alamos, p. 76-86, nov. 1990.

[Arm94] ARMSTRONG, C. G. Modelling requirements for finite element analysis. Computer aided design, Guildford, v. 26, n. 7, july 1994.

[Aut92] AUTODESK, Inc. Advanced Modeling Extension: Release 2, Reference Manual, Publication AC11AME2-02, 1992.

[Aut92a] AUTODESK, Inc. AutoCAD Development System: Programmer's Reference. Publication 100192-01, 1992.

[Ayr76] AYRES Jr, F. Geometria analítica plana e sólida. São Paulo: McGraw-Hill, 1976. (Coleção mini-Schaum, v. 3)

[Bad78] BADLER, N., BAJCSY, R. Three dimensional representations for computer graphics and computer vision. Computer graphics, New York, v. 12, n.3, p.153-160, 1978.

[Bae79] BAER, A., EASTMAN, C., HENRION, M. Geometric modelling: a survey. Computer aided design, Guildford, v. 11, n. 5, p. 253-272, sept. 1979.

[Bau75] BAUMGART, B. G. A Polyhedron representation for computer vision. In: NATIONAL COMPUTER CONFERENCE, 44, 1975. Proceedings..., p. 589-596.

[Boe94] BOENDER, E., BRONSVOORT, W. F., POST, F. H. Finite element mesh generation from constructive-solid-geometry models. Computer aided design, Guildford, v. 26, n. 5, p. 379-392, may 1994.

[Boo91] BOOCH, G. Object oriented design with applications. Benjamin/Cummings, 1991.

[Boo00] BOOCH, G., RUMBAUGH, J., JACOBSON, I. The unified modeling language user guide. Massachusetts: Addison Wesley, 2000. 482 p.

[Bou87] BOULOS, P., CAMARGO. I. Geometria analítica: um tratamento vetorial. 2 ed. São Paulo: McGraw-Hill, 1987. 385 p.

[Bra80] BRAID, I. C., HILLYARD. R. C., STROUD, I. A. Stepwise construction of polyedra in geometric modeling. In: BRODIE, K. W. (ed.) Mathematical methods in computer graphics and design. London: Academic, 1980. p. 123-141.

[Cal97] CALVERT, C. Borland C++ Builder unleashed. Sams, 1997. 1285 p.

[Car78] CAROLI, A. et al. Matrizes, vetores e geometria analítica. 9 ed. São Paulo: Nobel, 1978. 167 p.

[Cas90] CASACURTA, A., LASCHUK, A. Sweeping: representação de sólidos rígidos. In: JORNADA EPUSP/IEEE EM COMPUTAÇÃO VISUAL 1, 1990, São Paulo. Anais... São Paulo: EPUSP, 1990. p. 277-289.

[Cen92] CENSI, A. L. C., LADEIRA, M. C. AutoCad: release 11. 2 ed. São Paulo: Érica, 1992. 410 p.

[Chi85] CHIOKURA, H., KIMURA, F. A method of representing the solid design process. IEEE computer graphics and applications, Los Alamitos, v. 5, n. 4, p. 32-41, apr. 1985.

[Cho89] CHOI, Y., Vertex-based boundary representation on non-manifold geometric models. Ph. D. Thesis, Carnegie Mellow University, 1989.

[Coa96] COAD, P., YOURDON, C. Análise baseada em objetos. Rio de Janeiro: Campus, 2ª ed., 1996. 225 p.

[Col97] COLYER, B. et al. Project MIDAS: Magnet Integrated Design and Analysis System, IEEE transactions on magnetics, vol. 33, n. 2, pp. 1143-1148, mar. 1997.

[Del75] DEL MONACO, G., RE, V. Desenho eletrotécnico e eletromecânico. São Paulo, Hemus, 1975. 511 p.

[Des92] DESAULNIERS, H., STEWART, N. F. An extension of manifold boundary representations to the r-sets. ACM transactions on graphics. New York, v. 11, n. 1, p. 40-60, jan. 1992.

[Dob88] DOBKIN, D., GUIBAS, L., HERSHBERGER, J. e SNOEYNIK, J. An efficient algorithm for finding the CSG representation of a simple polygon. Computer graphics, New York, v. 22, n. 4, pp 31-40, 1988.

[Dol85] DOLCE, O., POMPEO, J. N. Fundamentos de matemática elementar. 6 ed. São Paulo: Atual, 1985. 10 v. (v.9: Geometria plana).

[Dol85a] DOLCE, O., POMPEO, J. N. Fundamentos de matemática elementar. 6 ed. São Paulo: Atual, 1985. 10 v. (v. 10: Geometria espacial - posição e métrica).

[Eas79] EASTMAN, C., WEILER, K. Geometric modeling using the Euler operators. In: ANNUAL CONF. COMPUTER GRAPHICS IN CAD/CAM SYSTEMS, 1, 1979, Cambridge. Proceedings... p. 248-254.

[Ell93] ELLIS, M. A., STROUSTRUP, B. C++ Manual de Referência Comentado. Rio de Janeiro: Campus, 1993.

[Fer86] FERREIRA, A. B. H. Novo dicionário da língua portuguesa. 2 ed. Rio de Janeiro: Nova Fronteira, 1986. 1838 p.

[Fil87] FILGUEIRAS, L. V. L. Fundamentos de computação gráfica. Rio de Janeiro: L.T.C., 1987. 356 p.

[Fis91] FISCHER, R. Genesys: sistema híbrido para modelagem de sólidos. Rio de Janeiro: Laboratório de CAD inteligente - Depto de Informática, 1991. 127 p. (Dissertação, Mestrado em Ciência da Computação). Pontifícia Universidade Católica do Rio de Janeiro - PUC, 1991.

[Fol90] FOLEY, J. D. et al. Computer graphics: principles and practice. 2 ed. Massachusetts: Addison Wesley, 1990. 1174 p.

[For90] FORDES, B. W. R., FOSCHI, R. O., STEIMER, S. F. Object-oriented finite element analysis, Comput. Struct. , V. 34, n. 3, pp. 355-374, 1990.

[Gam95] GAMMA, E., HELM, R., JOHNSON, R. VLISSIDES, J. Design patterns: elements of reusable object-oriented software. Massachusetts: Addison Wesley, 1995. 395 p.

[Gan83] GANE, C., SARSON, T. Análise estruturada de sistemas. Rio de Janeiro: L.T.C., 1983. 257 p.

[Gar97] GARNY, A. OpenGl component, software and documentation available from <http://pc-heartbreak.physiol.ox.ac.uk /programming_builder.html>.

[Gar00] GARTNER GROUP, INC. Adopting Linux Without Abandoning Windows. Borland Inprise Webletter: Gartner's Unix and Midrange Strategies Research Note DF-10-1058, February 2000. Available from <http://www.gartner.com/webletter/inprise/index1.html>.

[Geo91] GEORGE, P. L. Automatic mesh generation: application to finite element methods, John Wiley & Sons, 1991.

[Gom90] GOMES, J. M., VELHO, L. C. Conceitos básicos de computação gráfica. São Paulo: VII Escola de Computação, 1990. 311 p.

[Gon98] GONZALEZ, M. L. Solução de problemas magnetostáticos 3D com malhas de elementos finitos adaptativas - triangulação de Delaunay e versão H. Belo Horizonte: PPGEE-UFMG, 1998. (Exame de Qualificação, Doutorado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica - UFMG, 1998.

[Gon00] GONZALEZ, M. L. Geração adaptativa de malha de elementos finitos em três dimensões aplicada ao eletromagnetismo. Belo Horizonte: PPGEE-UFMG, 2000. (Tese, Doutorado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica - UFMG, 2000.

[Gon00a] GONZALEZ, M. L., MESQUITA, R. C., REBELO, G. C. Vimesh3D – um programa computacional para visualização de malhas de elementos finitos tridimensionais. Artigo submetido ao CILAMCE’2000.

[Gop96] GOPAC - Grupo de Otimização e Projeto Assistidos por computador. Definição de arquivo neutro para intercomunicação de dados em programas de cálculo de campos eletromagnéticos. Belo Horizonte: CPDEE-UFMG. Relatório Interno, 1996.

[Gop99] GOPAC - Grupo de Otimização e Projeto Assistidos por computador. Definição da base de dados neutra para intercomunicação de dados em programas de cálculo de campos eletromagnéticos – versão 4.0. Belo Horizonte: CPDEE-UFMG. Relatório Interno, 1999.

[Gu95] GU, P. CHAN, K. Product modelling using STEP. Computer aided design, Guildford, v. 27, n. 3, p. 163-179, mar. 1995.

[Gur91] GURSOZ, E. L., CHOI, Y., PRINZ, F. B. Boolean set operations on non-manifold boundary representation objects. Computer aided design, Guildford, v. 23, n. 1, p. 33-39, jan./ feb. 1991.

[Han96] HAN, Y. et al. A framework in developing kernel modeler based on nonmanifold boundary representation. In: PACIFIC CONFERENCE ON MANUFACTURING, 1996. Proceedings..., vol. 1, pp. 195-200.

[Hea86] HEARN, D., BAKER, M. P. Computer graphics. New Jersey: Prentice-Hall, 1986. 352 p.

[Hof87] HOFFMANN, C. M., HOPCROFT, J. E. Geometric ambiguities in boundary representations. Computer-aided design, Guildford, v. 19, n. 3, p. 141-147, apr. 1987.

[Hof89] HOFFMANN, C. M. Geometric and solid modeling. San Mateo: Morgan Kaufmann, 1989. 338 p.

[Hui92] HUI, K. C., TAN, S. T. Construction of a hybrid sweep-CSG modeler: the sweep-CSG representation. Engineering with computers, New York, v. 6, n. 2, p. 101-119, 1992.

[Iez85] IEZZI, G. Fundamentos de matemática elementar. 6 ed. São Paulo: Atual, 1985. 10 v. (v. 3: Trigonometria).

[Iez85a] IEZZI, G., HAZZAN, S. Fundamentos de matemática elementar. 6 ed. São Paulo: Atual, 1985. 10 v. (v. 4: Seqüências matrizes determinantes sistemas).

[Iez85b] IEZZI, G. Fundamentos de matemática elementar. 6 ed. São Paulo: Atual, 1985. 10 v. (v. 7: Geometria analítica).

[ISO91] ISO (INTERNATIONAL STANDARD ORGANIZATION) . Express Language Reference Manual, ISO TC184/SC4/WG5-n14, April 1991.

[Jar85] JARED, G. Boundary representations: data structures, computations, operations and applications. In: ANNUAL CONF. OF THE SPECIAL INTEREST GROUP ON COMPUTER GRAPHICS OF THE ASSOCIATION FOR COMPUTING MACHINERY, 12, 1985. Tutorial notes... New York: ACM, p.1-27.

[Jud71] JUDICE, E. D. Elementos de geometria analítica. 2 ed. Belo Horizonte: Vega, 1971. 298 p.

[Kam91] KAMEL, H. A., CHEN, L. Integration of solid modeling and finite element generation. Computer methods in applied mechanics and engineering, North-Holland, 89, p. 485-496, 1991.

[Kin76] KINDLE, J. H. Geometria analítica plana e no espaço. São Paulo: McGraw-Hill, 1976. (Coleção Schaum).

[Koe96] KOENIG, A. , MOO, B. Ruminations on C++: a decade of programming insight and experience. Addisson-Wesley, USA, 1996. 380 p.

[Kro89] KROSZYNSKIU., PALSTROEM, B., TROSTMANN, E., SCHLECHTENDAHL, G. Geometric data transfer between CAD systems: solid models. IEEE computer graphics and applications, Los Alamitos, p. 57-71, sept. 1989.

[LaC95] LACOURSE, D. E. Handbook of solid modeling. New York: McGrawHill, 1995.

[Lea79] LEAN, M. H., FRIEDMAN, M., WEXLER, A. Application of the boundary element method in electrical engineering problems. In: BANERJEE, P. K., BUTTERFIELD, R.(ed.) Developments in boundary element methods – 1. London: Applied Science Publishers, 1979. p. 207-250.

[Lee93] LEE, S. Feature-based non-manifold geometric modelling system to provide integrated environment for design and analysis of injection molding products. Ph. D. Thesis, Seoul National University, 1993.

[Leh79] LEHMAN, C. H. Geometria analítica. 3 ed. Porto Alegre: Globo, 1979.

[Lei77] LEITHOLD, L. O cálculo com geometria analítica. São Paulo: Harper & Row do Brasil, 1977. v. 2.

[Mag87] MAGNENAT-THALMAN, N. THALMAN, D. Image synthesis: theory and practice. Tokyo: Springer-Verlag, 1987. 400 p.

[Mag94] MAGALHÃES, A. L. C. C. Um modelador de sólidos multirrepresentacional: estudo, projeto e implementação. São Carlos: ICMSC-USP, 1994. (Dissertação, Mestrado em Ciência da Computação), Instituto de Ciências Matemáticas de São Carlos - USP, 1994.

[Mag94a] MAGALHÃES, A. L. C. C. Operadores de Euler na modelagem de sólidos por fronteira: conceito, aplicação, estudos de casos. São Carlos: ICMSC-USP, 1994. 29 p. (Notas Técnicas do ICMSC).

[Mag94b] MAGALHÃES, A. L. C. C. et al. Desenvolvimento de um modelador de sólidos multirrepresentacional com núcleo B-rep e técnicas de descrição por varredura e semi-espaços. In: SIBGRAPI - SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 7, 1994, Curitiba. Comunicações... p. 13-16.

[Mag96] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Solid modeling application in electromagnetism. In: CBMAg - CONGRESSO BRASILEIRO DE ELETROMAGNETISMO, 2, 1996, Ouro Preto. Anais... p. 147-150.

[Mag96a] MAGALHÃES, A. L. C. C. AutoPAC versão 2.0 - documentação de projeto e implementação. Belo Horizonte: CPDEE-UFMG. Relatório Interno, 1996.

[Mag97] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Requirements for a solid modeler coupled to a finite element mesh generators. In: COMPUMAg - CONFERENCE ON THE COMPUTATION OF ELECTROMAGNETIC FIELDS, 11, 1997, Rio de Janeiro. Proceedings... p. 319-320.

[Mag98] MAGALHÃES, A. L. C. C. Estudo, projeto e implementação de um modelador de sólidos voltado para aplicações em eletromagnetismo. Belo Horizonte: PPGEE-UFMG, 1998. (Exame de Qualificação, Doutorado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica - UFMG, 1998.

[Mag98a] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Requirements for a solid modeler coupled to a finite element mesh generator, IEEE transactions on magnetics, pp. 3347-3350 , September 1998.

[Mag98b] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Tratamento de fronteiras internas em modeladores de sólidos voltados para o eletromagnetismo. In: CBMAg - CONGRESSO BRASILEIRO DE ELETROMAGNETISMO, 3, 1998, São Paulo. Anais. p. 250-253.

[Mag98c] MAGALHÃES, A. L. C. C. Essa tal "compugrafia". CADTEC Mídia, Belo Horizonte, n.5, p. 3-4, outubro de 1998.

[Mag99] MAGALHÃES, A. L. C. C. Apresentação bidimensional de sólidos. CADTEC Mídia, Belo Horizonte, n.6, p. 3-4, fevereiro de 1999.

[Mag99a] MAGALHÃES, A. L. C. C. Representação computacional de sólidos. CADTEC Mídia, Belo Horizonte, n.7, p. 3-4, junho de 1999.

[Mag99b] MAGALHÃES, A. L. C. C et. al. Desenvolvimento de um modelador de sólidos voltado para aplicações em eletromagnetismo. In: Semana da Pós-Graduação da Universidade Federal de Minas Gerais, 1, 1999, Belo Horizonte. Anais. p.112. (Resumo) - Premiado como um dos melhores trabalhos na Área de Engenharia apresentados no evento.

[Mag99c] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Exploring inner boundaries in solid modelers applied to electromagnetic problems. In: COMPUMAg - CONFERENCE ON THE COMPUTATION OF ELECTROMAGNETIC FIELDS, 12, 1999, Sapporo - Japão. Proceedings... p. 214-215.

[Mag99d] MAGALHÃES, A. L. C. C., SHIMIZU, C. A., MESQUITA, R. C. Uma estratégia para gerar malha de elementos finitos em sólidos definidos por varredura. In: CILAMCE – CONGRESSO IBERO LATINO AMERICANO DE MÉTODOS COMPUTACIONAIS EM ENGENHARIA, 20, 1999, São Paulo. Abstracts... p. 272, CD-ROM – publicação 183, 11 p.

[Mag99e] MAGALHÃES, A. L. C. C. et al. Um modelador de sólidos voltado para aplicações em eletromagnetismo. In: CILAMCE – CONGRESSO IBERO LATINO AMERICANO DE MÉTODOS COMPUTACIONAIS EM ENGENHARIA, 20, 1999, São Paulo. Abstracts... p. 262, CD-ROM – publicação 184, 20 p.

[Mag99f] MAGALHÃES, A. L. C. C. Tratamento de fronteiras internas em modeladores de sólidos. CADTEC Mídia, Belo Horizonte, n.7, p. 3-4, dezembro de 1999.

[Mag00] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Exploring inner boundaries in solid modelers applied to electromagnetic problems. Aceito para publicação na edição do periódico IEEE transactions on magnetics alusivo ao CompuMag'99.

[Mag00a] MAGALHÃES, A. L. C. C., MESQUITA, R. C. Construção da fronteira em sólidos definidos por varredura. Artigo submetido ao CBMag'2000.

[Män82] MÄNTYLÄ, M., SULONEN, R. GWB: a solid modeler with Euler operators. IEEE computer graphics and applications, Los Alamitos, v. 2, n. 7, p. 17-31, sept. 1982.

[Män83] MÄNTYLÄ, M., TAMMINEN, M. Localised set operations for solid modeling. Computer graphics, New York, v. 17, n. 3, p. 279-288, jul. 1983.

[Män84] MÄNTYLÄ, M. A note on the modeling space of Euler operators. Computer vision, graphics and image processing, San Diego, v. 26, p. 45-60, 1984.

[Män86] MÄNTYLÄ, M. Boolean operations of 2-manifolds through vertex neighborhood classification. ACM transactions on graphics, New York, v. 5, n. 1, p. 1-29, 1986.

[Män88] MÄNTYLÄ, M. An introduction to solid modeling. Maryland: Computer Science, 1988. 401 p.

[Mar87] MARSHALL, G. R. Computer graphics in application. New Jersey: Prentice Hall, 1987. 454 p.

[Mas93] MASUDA, H., Topological operators and boolean operations for complex-based nonmanifold geometric models. Computer aided design, Guildford, v. 25, n. 2, p. 119-129, 1993.

[Mes97] MESQUITA, R. C., SOUZA, R. P., PINHEIRO, T., MAGALHÃES, A. L. C. C. An object-oriented platform for teaching finite element pre-processor programming and design techniques. In: COMPUMAg - CONFERENCE ON THE COMPUTATION OF ELECTROMAGNETIC FIELDS, 11, 1997, Rio de Janeiro. Proceedings... p.191-192.

[Mes98] MESQUITA, R. C., SOUZA, R. P., MAGALHÃES, A. L. C. C. An object-oriented platform for teaching finite-element pre-processor programming and design techniques, IEEE transactions on magnetics, pp. 3407-3410 , September 1998.

[Mia97] MIANO, J., CABANSKI, T., HOWE, H. C++ Builder how to. Corte Madeira: Waite Group Press, 1997. 822 p.

[Mil89] MILLER, J. R. Arquitectural issues in solid modelers. IEEE computer graphics and applications, Los Alamitos, p. 72-87, sept. 1989.

[Mon94] MONTENEGRO, P., PACHECO, R. Orientação a objetos em C++ . Rio de Janeiro: Ciência Moderna, 1994. 394 p.

[Mor85] MORTENSON, M. E. Geometric modeling. New York: John Wiley, 1985. 763 p.

[Mus96] MUSSER, D. L., SAINI, A. STL tutorial and reference guide. Massachusetts: Addison Wesley, 1996.

[Muu91] MUUSS, M. J., BUTLER, L. A. Combinatorial solid geometry, boundary representations and non-manifold geometry, State of the art in computer graphics: vizualization and modeling. New York: Springer-Verlag,1991.

[Nun00] NUNES, C. R. S. Implementação de operações booleanas em um modelador de sólidos voltado para aplicações em eletromagnetismo. (Plano de Dissertação, Mestrado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica - UFMG, 2000.

[Oli91] OLIVEIRA, M. C. F. Modelagem geométrica: uma introdução. São Carlos: Instituto de Ciências Matemáticas de São Carlos - Universidade de São Paulo. Relatório Interno, 1991.

[Owe97] OWEN, J. STEP: an introduction. 2nd ed. Winchester: Information Geometers, 1997. 203p.

[Pèl92] PÈLERIN, Y. D., ZIMMERMANN, T., BOMME, P. Object-oriented finite element programming II: a prototype program in Smaltalk, Computer methods in applied and mechanical engineering, v. 98, n. 3, pp. 361-397, 1992.

[Per90] PERSIANO, R. C. M., OLIVEIRA, A. A. F. Introdução à computação gráfica. Rio de Janeiro: L.T.C., 1990.

[Per94] PERRY, G. Programação orientada para objeto com Turbo C++ . Rio de Janeiro: Berkeley, 1994. 752 p.

[Pet96] PETZOLD, C., YAO, P. Programming Windows 95. Washington: Microsoft Press, 1996.

[Pil89] PILZ, M., KAMEL, H. A. Creation and boundary evaluation of CSG-Models. Engineering with computers, New York, v. 5, n. 2, p. 105-118, 1989.

[Pra90] PRATT, M. J. Solid modeling - survey and current research issues. In: ROGERS, D. F., EASTMAN, R. A. (ed.). Computer graphics techniques: theory and practice. New York: Springer-Verlag, 1990. p. 363-405.

[Pre87] PRESSMAN, R. S. Software engineering: a practitioner's approach. 2. ed. Singapore: McGraw-Hill, 1987. 567 p.

[Qua98] QUATRANI, T. Visual modeling with Rational Rose and UML. Addison Wesley, USA, 1998.

[Raz82] RAZEVIC, D. V. High Voltage Engineering. Trad. M. P. Chourasia. Delhi: Khanna Publishers, 1982. 726 p.

[Rei98] REISDORPH, K. Teach yourself Borland C++ Builder 3 in 21 days. Indianapolis: Sams Publishing, 1998. 832 p.

[Req77] REQUICHA, A. G., VOELCKER, H. B. Constructive solid geometry. Rochester: University of Rochester, 1977. (Production Automation Project -Tech. Memo., 25).

[Req80] REQUICHA, A. G. Representations for rigid solids: theory, methods and systems. ACM computing surveys, New York, v. 12, n. 4, p. 437-464, dec. 1980.

[Req82] REQUICHA, A. G., VOELCKER, H. B. Solid modeling: a historical summary and contemporary assessment. IEEE computer graphics and applications, Los Alamitos, v. 2, p. 9-24, mar. 1982.

[Req83] REQUICHA, A. G., VOELCKER, H. B. Solid modeling: current status and research directions. IEEE computer graphics and applications, Los Alamitos, v. 3, p. 25-37, oct. 1983.

[Req85] REQUICHA, A. G., VOELCKER, H. B. Boolean operations in solid modeling: boundary evaluation and merging algorithms. Proceedings of the IEEE, New York, v. 73, n. 1, p. 30-44, jan. 1985.

[Req92] REQUICHA, A. G., ROSSIGNAC, J. R. Solid modeling and beyond. IEEE computer graphics and applications, Los Alamitos, p. 31-44, sept. 1992.

[Ric72] RICH, B. Geometria plana. São Paulo: McGraw-Hill, 1972. (Coleção Schaum). 312 p.

[Roc95] ROCHA, L. F. N., MESQUITA, R. C. Uma base de dados de formato neutro para intercomunicação entre programas de cálculo de campos eletromagnéticos: versão 1.1, In: CBMAg - CONGRESSO BRASILEIRO DE ELETROMAGNETISMO, 1, 1995, Florianópolis. Anais... , pp: 140-143.

[Roc96] ROCHA, L.F.N.; MESQUITA, R.C. An object-oriented data structure for a 3-D electromagnetic field program preprocessor. IEEE Transactions on Magnetics, v. 32, n. 3, pp. 1449-1452, may 1996.

[Rog90] ROGERS, D. F., ADAMS, J. A. Mathematical elements for computer graphics. 2a ed. São Paulo: McGraw-Hill, 1990. 611 p.

[Ros90] ROSSIGNAC, J., O'CONNOR, M. A. SGC: a dimensional-independent model for pointsets with internal structures and incomplete boundaries. Geometric modeling for product engineering. North Holland, 1990. pp 145-180.

[Rum97] RUMBAUGH, J.et al. Modelagem e projetos baseados em objetos. Trad. Dalton Conde de Alencar. Rio de Janeiro: Campus, 1997. 652 p.

[Sab93] SABONNADIÈRE, J. C., COULOMB, J. L. Elementos finitos e CAE - aplicações em engenharia elétrica. São Paulo: Aleph, 1993. 214 p.

[Sad93] SADI, R.A., COLYER, B., EMSON, C.R.I., SIMKIN, J., MAANEN, J.V. A 2D and axisymmetric finite element environment based upon STEP-type database. In: COMPUMAg - CONFERENCE ON THE COMPUTATION OF ELECTROMAGNETIC FIELDS, 9, 1993, Miami. Proceedings..., pp. 562-563.

[San98] SANTOS, R. J. Geometria analítica e álgebra linear. Disponibilidade e acesso <http://www.mat.ufmg.br/~regi/gaaltt/gaaltt.html>

[Seg90] SEGAL, M. Using tolerances to guarantee valid polihedral modeling results. Computer graphics, New York, v. 24, n. 4, p. 105-114, aug. 1990.

[Sep67] SEPÚLVEDA, H. L., CUNHA, E. M. Transformadores estáticos – cálculo dos transformadores. Belo Horizonte: Edições Engenharia / UFMG, 1967.

[Sha91] SHAPIRO, V. R e VOSSLER, D. L. Construction and optimization of CSG representations. Computer-aided design, Guildford, v. 23, n. 1, p. 4-20, january / february 1991.

[Sha95] SHAH, J. J., MÄNTYLA, M. Parametric and feature-based CAD / CAM: concepts, techniques and applications. New York: John Wiley, 1995. 619 p.

[She85] SHEPHARD, M. S. Finite element modelling within an integrated geometric modelling environment. Part II: Attribute specification, domain differences, and indirect element types. Engeneering with computers, v. 1, p. 73-85, 1985.

[She96] SHEWCHUK, J. R. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: WORKSHOP ON APPLIED COMPUTATIONAL GEOMETRY, I, 1996, Philadelphia. Proceedings... ACM, 1996. p. 124-133. Available from <http:://www.cs.cmu.edu/afs/cs/project/quake/public/www/tripaper/triangle0.html>; software available from <http:://www.cs.cmu.edu/~quake/triangle.html>

[Shi83] SHIRMA, Y., OKINO, N., KAKUZU, Y. Research on 3D geometric modeling by sweep primitives. In: COMPUTER AIDED DESIGN, 1983. Proceedings... p. 671-680.

[Sil93] SILVA, E. J., MESQUITA, R .C., SALDANHA, R. R., PALMEIRA, P. F. M. An object-oriented finite element program for electromagnetic field computation. In: COMPUMAG - CONFERENCE ON THE COMPUTATION OF ELECTROMAGNETIC FIELDS, 9, Miami, 1993. Proceedings... pp:236-237.

[Sil94] SILVEIRA Jr., L. G., TING, W. S. Operadores booleanos para a classe de NM-conjuntos. In: SIBGRAPI - SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 7, 1994, Curitiba. Anais... p. 271-278.

[Sil96] SILVA, E.J.; MESQUITA, R.C. Data management in finite element analysis programs using object-oriented techniques. IEEE Transactions on Magnetics, v. 32, n. 3, pp. 1445-1448, may 1996.

[Sol94] SOLANO, L., BRUNET, P. Constructive constraint-based model for parametric cad systems, Computer aided design, Guildford, v. 26, p. 614-621, August 1994.

[Sta98] STAA, A. V. Laboratório de Programação. Disponibilidade e acesso <ftp://genesis.les.inf.puc-rio.br/pub/courses/labprog1/>

[Str97] STROUSTROUP, B. The C++ Programing language. Addison Wesley, USA, 1997. 911 p.

[Tho95] THOMAS, D., GREENOUGH, C. Data modeling for electromagnetic and stress analysis integration. Rutherford Appleton Lab., March 1995.

[Til80] TILOVE, R. B., REQUICHA, A. A. G., Closure of Boolean operations on geometric entities. Computer aided design, v. 12, n. 5, pp. 219-220, sept. 1980.

[Til80a] TILOVE, R. B. Set membership classification: a unified approach to geometric intersection problems. IEEE transactions on computers, Los Alamitos, p. 874-883, oct. 1980.

[Tor84] TOR, S. B. e MIDDLEDITCH, A. E. Convex decomposition of simple polygons. ACM transactions on graphics, New York, v. 3, n. 4, pp 244-265, 1984.

[Tor86] TORIYA, H. et al. Undo and redo operations for solid modeling. IEEE computer graphics and applications, Los Alamitos, p. 35-42, apr. 1986.

[Tsu92] TSUZUKI, M. S. G. MSD - Modelador de sólidos didático. In: SIBGRAPI - SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 5, 1992, Águas de Lindóia. Comunicações... p. 17-20.

[Tur88] TURNER, J. U. Accurate solid modeling using polyhedral approximations. IEEE computer graphics and applications, Los Alamitos, p. 14-28, may 1988.

[Voe77] VOELCKER, H. B., REQUICHA, A. G. Geometric modeling of mechanical parts and processes. Computer, New York, p. 48-57, dec. 1977.

[Wei87] WEILER, K. J. The radial edge structure: a topological representation for non-manifold geometric modeling. In: WOZNY, M., MCLAUGHLIM, H., ENCARNACAO, J. (ed.), Geometric modeling for CAD applications, New York: Springer-Verlag, 1987.

[Wie91] Wiener, R. S., Pinson, L. J. Programação orientada para objeto e C++. São Paulo: Makron Books do Brasil, 1991.

[Wil85] WILSON, P. Euler Formulas and geometric modeling. IEEE computer graphics and applications. Los Alamitos, pp 24-36, August 1985.

[Wil87] WILSON, P. R. A short history of CAD data transfer standards. IEEE computer graphics and applications, Los Alamitos, p. 64-67, jun. 1987.

[Wol84] WOLFGRAM, D. E. Aventuras em 3D . Rio de Janeiro: Berkeley, 1993.

[Wri96] WRIGHT, R. OpenGL Superbible. Corte Madera: Waite Group, 1996.

[Yam91] YAMAGUSHI, Y., KOBAYASHI, K., KIMURA, F. Geometric modeling with generalised topology and geometry for product engineering. In: TURNER, J., PENGNA, J., WOZNY, M (ed.) Product modeling for computer-aided design and manufacturing. North-Holland: Elsevier Science Publishers, 1991, pp. 97-115.

[Zim92] ZIMMERMANN, T., PÈLERIN, Y. D., BOMME, P. Object-oriented finite element programming: I governing principles. Computer methods in applied and mechanical engineering, v. 98, n. 2, pp. 291-303, 1992.